The IoT Hacker’s Handbook

The IoT Hacker’s Handbook

Um guia prático para Hacking da internet das coisas

**Introdução**

Os dez capítulos deste livro abrangem uma série de tópicos, desde exploração de hardware e firmware embarcado até comunicação por rádio, incluindo exploração de BLE e ZigBee.

Para mim, escrever este livro foi uma jornada emocionante e cheia de aventura, compartilhando minhas experiências e as várias coisas que aprendi em minha carreira profissional e colocando tudo nesses dez capítulos.

Espero que você possa aproveitar ao máximo este livro e eu altamente te encorajo a pegar todos os conjuntos de habilidades aprendidos aqui e aplicá-los a problemas do mundo real, ajudando a tornar o ecossistema da Internet das Coisas (IoT) mais seguro. São as contribuições individuais que nos ajudarão a criar um mundo mais seguro, e você, que está lendo este livro, pode fazer parte disso.

Ninguém é perfeito, e este livro certamente terá um ou dois pequenos erros. Se você encontrar algum desses erros, por favor, me avise e ficarei feliz para corrigi-los em futuras edições do Manual do Hacker da IoT.

Eu também dou aulas de três e cinco dias sobre exploração de IoT ofensiva, que eu recomendo que você participe para obter experiência prática com tudo o que é abordado no livro. Para mais informações sobre o treinamento online e as aulas ao vivo, acesse attifystore.com.

A última e mais importante parte é a comunidade! Para você, leitor, eu quero que esteja disposto o suficiente para compartilhar seu conhecimento com seus colegas ou mesmo com alguém que é novo nesta área. É assim que nós, como comunidade, cresceremos.

É tudo da minha parte. Mais uma vez, obrigado por ler o Manual do Hacker da IoT e desejo a todos o melhor em seus empreendimentos de exploração de IoT.

Aditya Gupta (@adi1391)

Fundador e Chefe Hacker,

Attify

**Agradecimentos**

Esse livro nunca teria sido terminado sem o meu time incrível da Attify, que trabalharam dia e noite para ter certeza que produzimos um conteúdo de qualidade como uma equipe.

**1.Internet das coisas: Um primer**

No mundo da tecnologia da comunicação, dois dos eventos que possuem significância especial são a invenção da ARPANET, uma rede de computadores que permite a troca de dados entre computadores mesmo quando estão geograficamente separados, e a ascensão da Internet das Coisas (IoT). Este último, no entanto, foi um processo evolutivo ao invés de um único evento.

As primeiras implementações do conceito de IoT ocorreram quando dois estudantes da Universidade Carnegie Mellon encontraram uma maneira de monitorar o número de latas restantes em uma máquina de venda automática permitido que dispositivos se comuniquem com o mundo externo. Eles fizeram isso adicionando um fotossensor ao dispositivo que contaria toda vez que uma lata saía da máquina, e assim, o número de latas restantes era calculado. Hoje em dia, os dispositivos IoT são capazes de monitorar sua frequência cardíaca e até mesmo controlá-la, se necessário, em caso de um evento adverso. Além disso, alguns dispositivos IoT agora podem servir como fonte de evidência durante julgamentos em tribunais, como visto no final de 2015, quando os dados do FitBit de uma mulher foram usados em um julgamento de assassinato. Outros incidentes incluem o uso de dados de marcapasso e gravações do Amazon Echo em vários julgamentos. A jornada dos dispositivos IoT de um dormitório universitário para dentro de seres humanos é fascinante, para dizer o mínimo. Kevin Aston, quando mencionou o termo Internet das Coisas pela primeira vez, provavelmente não imaginaria que esses dispositivos logo ultrapassariam a população humana total em número. Aston mencionou o termo em referência à tecnologia de identificação por radiofrequência (RFID), que estava sendo usada para conectar dispositivos entre si. A definição de IoT mudou desde então, com diferentes organizações dando seu próprio significado ao termo. A Qualcomm e a Cisco criaram o termo Internet of Everything (IoE), que alguns acreditam ser para uma agenda de marketing. O termo, de acordo com eles, significa estender o conceito de IoT de ser limitado à comunicação máquina-a-máquina para máquinas se comunicando com máquinas e com o mundo físico.

O primeiro vislumbre da IoT atual foi visto em junho de 2000, quando a primeira geladeira conectada à Internet, a Internet Digital DIOS, foi revelada pela LG. A geladeira continha uma tela TFT-LCD de alta qualidade com uma série de funcionalidades, incluindo exibir a temperatura dentro da geladeira, fornecer pontuações de frescura dos itens armazenados e usar a funcionalidade da webcam para rastrear os itens armazenados. O dispositivo inicial que provavelmente chamou mais atenção da mídia e dos consumidores foi o Nest Learning Thermostat em outubro de 2011. Este dispositivo era capaz de aprender a programação do usuário para ajustar diferentes temperaturas desejadas em diferentes horas do dia. A aquisição desta empresa de termostato IoT pelo Google por US$ 3,2 bilhões foi o evento que conscientizou o mundo sobre a revolução tecnológica que se aproximava.

Logo, havia centenas de novas startups tentando interconectar todos os diferentes aspectos do mundo físico a dispositivos e grandes organizações criando equipes internas especializadas para criar sua própria linha de dispositivos IoT para serem lançados no mercado o mais rápido possível. Essa corrida para criar novos dispositivos chamados de inteligentes nos leva ao presente, onde podemos interagir com nossas smart TVs em casa enquanto tomamos uma xícara de café preparada por uma máquina de café controlada pela Internet e controlamos as luzes pela música tocando em seu assistente inteligente. A IoT, no entanto, não se limita apenas ao nosso espaço físico. Ele também possui inúmeras aplicações em empresas, lojas de varejo, saúde, indústria, redes de energia e até mesmo em pesquisas científicas avançadas.

Os formuladores de políticas do mundo digital lutaram contra o ritmo acelerado do surgimento de dispositivos IoT e não conseguiram criar controles de qualidade e regulamentações de segurança rígidos. Isso só mudou recentemente, quando organizações como a GSMA criaram diretrizes de segurança e privacidade para dispositivos IoT, e a Comissão Federal de Comércio (FTC) Foram definidas etapas a serem seguidas para garantir a segurança. No entanto, a demora levou à adoção generalizada de dispositivos de IoT em todos os setores, e também permitiu que os desenvolvedores ignorassem as considerações de segurança no que diz respeito a esses dispositivos. Não foi até o efeito generalizado do botnet Mirai que as deficiências de segurança desses dispositivos se tornaram aparentes Dispositivos seriam vulneráveis. Mirai era uma botnet que atacava dispositivos de Internet das Coisas (IoT), principalmente câmeras conectadas à internet, verificando o status das portas 23 e 2323 e forçando a autenticação por força bruta usando credenciais comuns. Sem surpresa, muitas das câmeras IP expostas à internet tinham acesso telnet disponível com um nome de usuário e senha extremamente comuns, o que era fácil de encontrar. A mesma botnet também foi usada posteriormente para assumir o controle da infraestrutura de internet da Libéria, bem como da DYN, o que levou a um ataque a vários sites populares, incluindo GitHub, Twitter, Reddit e Netflix.

Nos últimos anos, embora a segurança desses dispositivos tenha melhorado lentamente, ainda não atingiu um ponto onde eles possam ser considerados extremamente seguros de usar. Em novembro de 2016, quatro pesquisadores de segurança - Eyal Ronen, Colin O'Flynn, Adi Shamir e Achi-Or Weingarten - criaram um interessante worm de prova de conceito (PoC) que atacava usando drones e tomava o controle das luzes inteligentes Philips Hue de um prédio comercial. Mesmo que o ataque tenha sido apenas um PoC, não é exagero pensar que veríamos ransomware para dispositivos inteligentes semelhante ao WannaCry, pedindo dinheiro para abrir a fechadura da porta ou ligar um marcapasso. Quase todos os dispositivos inteligentes comprovadamente possuem problemas críticos de segurança e privacidade, incluindo sistemas de automação residencial inteligente, dispositivos vestíveis, monitores de bebês e até mesmo brinquedos sexuais pessoais. Considerando a quantidade de dados íntimos que esses dispositivos coletam, é assustador ver o quanto estamos expostos a ataques cibernéticos.

O aumento de incidentes de segurança em dispositivos IoT também levou a uma demanda crescente por profissionais de segurança IoT, atuando tanto na construção quanto na quebra desses sistemas. Isso permite que as organizações garantam que seus dispositivos estejam protegidos das vulnerabilidades que invasores mal-intencionados podem usar para comprometer seus sistemas. Além disso, várias empresas começaram a oferecer recompensas por bugs (bug bounties) para incentivar pesquisadores a avaliar a segurança de seus dispositivos IoT, algumas até mesmo enviando dispositivos físicos gratuitos para esses pesquisadores. Nos próximos anos, essa tendência deve crescer e, com o aumento das soluções de IoT no mercado, haverá uma demanda maior por profissionais especializados em segurança IoT no mercado de trabalho.

**Problemas anteriores de segurança de IoT**

A melhor forma de aprender sobre segurança desses dispositivos é olhar o que aconteceu no passado. Ao aprender sobre os erros de segurança cometidos por outros desenvolvedores de produtos no passado, podemos entender quais tipos de problemas de segurança devemos esperar no produto que estamos avaliando. Mesmo que alguns termos pareçam desconhecidos aqui, nós os discutiremos em detalhes nos próximos capítulos.

**Termostato Nest**

O artigo “Smart Nest Thermostat: Um Espião Inteligente em Sua Casa”, de Grant Hernandez, Orlando Arias, Daniel Buentello e Yier Jin, menciona algumas das deficiências de segurança do Google Nest que permitiam a instalação de um novo firmware malicioso no dispositivo. Isso era feito pressionando o botão do Nest por cerca de 10 segundos para acionar a reinicialização global. Nesse estágio, o dispositivo poderia ser configurado para procurar firmware em mídia USB comunicando-se com o pino sys\_boot5. Na unidade USB, estava presente um firmware malicioso, que o dispositivo então utilizava durante a inicialização.

Jason Doyle identificou outra vulnerabilidade nos produtos Nest que envolvia o envio de um valor personalizado nos detalhes do identificador de conjunto de serviços Wi-Fi (SSID) via Bluetooth para o dispositivo alvo, o que então travava o dispositivo e acabava por reiniciá-lo. Isso também permitiria a um ladrão invadir a casa durante a reinicialização do dispositivo (cerca de 90 segundos) sem ser flagrado pela câmera de segurança Nest.

**Casa inteligente Philips**

Dispositivos domésticos da Philips sofreram de uma série de problemas de segurança em toda a linha de produtos. Isso inclui o popular worm Philips Hue, criado como uma prova de conceito (PoC) pelos pesquisadores de segurança Eyal Ronen, Adi Shamir, Achi-Or Weingarten e Colin O'Flynn. No PoC, eles demonstraram como as chaves de criptografia simétricas embutidas usadas pelos dispositivos Philips poderiam ser exploradas para obter controle sobre os dispositivos alvo através do ZigBee. Também incluía a infecção automática de lâmpadas Philips Hue colocadas próximas umas das outras.

Em agosto de 2013, Nitesh Dhanjani, um pesquisador de segurança, também surgiu com uma nova técnica para causar blecautes permanentes usando uma técnica de ataque por replay para obter controle dos dispositivos Philips Hue. Ele descobriu essa vulnerabilidade depois de perceber que os dispositivos inteligentes Philips Hue consideravam apenas o MD5 do endereço de controle de acesso de mídia (MAC) como o único parâmetro para validar a autenticidade de uma mensagem. Como o invasor pode facilmente aprender o endereço MAC do host legítimo, ele pode criar um pacote malicioso indicando que veio do host genuíno e com os pacotes de dados com o comando para desligar a lâmpada. Fazer isso continuamente permitiria ao invasor causar um blecaute permanente, sem que o usuário tivesse outra opção a não ser substituir a lâmpada.

O Philips Hue (e muitos outros dispositivos inteligentes hoje) usa uma tecnologia chamada ZigBee para trocar dados entre os dispositivos, minimizando o consumo de recursos. O mesmo ataque que era possível no dispositivo usando pacotes Wi-Fi também seria aplicável ao ZigBee. No caso do ZigBee, tudo o que um invasor precisa fazer é simplesmente capturar os pacotes ZigBee para uma solicitação legítima e simplesmente reproduzi-los para realizar a mesma ação em um momento posterior e assumir o controle do dispositivo. Veremos também como capturar e reproduzir pacotes ZigBee no Capítulo 10.

**Lâmpada inteligente Lifx**

Dispositivos de casa inteligente têm sido um dos alvos de pesquisa mais populares entre a comunidade de segurança. Outro exemplo inicial ocorreu quando Alex Chapman, um pesquisador de segurança da empresa Context, descobriu sérias vulnerabilidades de segurança na lâmpada inteligente Lifx, tornando possível para invasores injetarem pacotes maliciosos na rede, obter credenciais WiFi descriptografadas e assumir o controle das lâmpadas inteligentes sem qualquer autenticação.

Os dispositivos, neste caso, estavam se comunicando usando 6LoWPAN, que é outro protocolo de comunicação de rede (assim como ZigBee) construído sobre o 802.15.4. Para farejar os pacotes 6LoWPAN, Chapman usou um Atmel RzRaven flashed com a imagem de firmware Contiki 6LoWPAN, permitindo que ele examinasse o tráfego entre os dispositivos. A maior parte da troca de dados confidenciais ocorrendo nessa rede era criptografada, o que fazia com que o produto parecesse bastante seguro.

Uma das coisas mais importantes durante o teste de penetração de IoT é a capacidade de olhar para o produto como um todo, em vez de apenas olhar para um único componente para identificar os problemas de segurança. Isso significa que para descobrir como os pacotes estão sendo criptografados na comunicação via rádio, a resposta provavelmente está no firmware. Uma das técnicas para obter o binário do firmware de um dispositivo é despejá-lo por meio de técnicas de exploração de hardware, como JTAG, que abordaremos no Capítulo 6. No caso das lâmpadas Lifx, o JTAG deu acesso ao firmware Lifx, que, quando revertido, levou à identificação do tipo de criptografia, que neste caso era o Advanced Encryption Standard (AES), a chave de criptografia, o vetor de inicialização e o modo de bloco usado para criptografia. Como essas informações seriam as mesmas para todas as lâmpadas inteligentes Lifx, um invasor poderia assumir o controle de qualquer lâmpada e invadir o Wi-Fi porque o dispositivo também estava comunicando as credenciais de Wi-Fi pela rede de rádio, que agora poderia ser descriptografada.

**O Hack do “Jeep”**

O Hack do Jeep é provavelmente o hack de IoT mais popular de todos os tempos. Dois pesquisadores de segurança, Dr. Charlie Miller e Chris Valasek, demonstraram em 2015 como poderiam assumir e controlar remotamente um Jeep usando vulnerabilidades no sistema Uconnect da Chrysler, resultando no recall de 1,4 milhão de veículos pela Chrysler.

O hack completo se aproveitou de muitas vulnerabilidades diferentes, incluindo extensos esforços na engenharia reversa de vários binários e protocolos individuais. Uma das primeiras vulnerabilidades que tornou o ataque possível foi o software Uconnect, que permitia a qualquer pessoa se conectar remotamente a ele por meio de uma conexão celular. A porta 6667 estava acessível com autenticação anônima habilitada e foi encontrada executando D-Bus sobre IP, que é usado para comunicação entre processos. Depois de interagir com o D-Bus e obter uma lista de serviços disponíveis, um dos serviços com o nome NavTrailService tinha um método de execução que permitia aos pesquisadores executar código arbitrário no dispositivo. A Figura 1-1 mostra o código de exploração usado para abrir um shell root remoto na unidade principal.

<http://illmatics.com/Remote%20Car%20Hacking.pdf>

Uma vez que a execução arbitrária de comandos foi obtida, tornou-se possível realizar um movimento lateral e enviar mensagens CAN assumindo o controle dos vários elementos do veículo, como volante, freios, faróis e assim por diante.

**Belkin Wemo**

Belkin Wemo é uma linha de produtos que oferece automação residencial completa aos consumidores.

Belkin Wemo é um caso interessante em que os desenvolvedores tomaram precauções para impedir que invasores instalassem firmware malicioso no dispositivo. As atualizações de firmware para Belkin Wemo, no entanto, aconteciam por um canal não criptografado, o que permitia aos invasores modificar o pacote binário do firmware durante a atualização. Como medida de proteção, o Belkin Wemo utilizava um mecanismo de distribuição de firmware criptografado baseado em GNU Privacy Guard (GPG) para que o dispositivo não aceitasse pacotes de firmware malicioso injetados por um invasor. Essa proteção de segurança foi superada com extrema facilidade porque o dispositivo estava distribuindo a chave de assinatura do firmware junto com o firmware durante o processo de atualização, tudo em um canal não criptografado. Um invasor poderia, portanto, modificar facilmente o pacote, bem como assiná-lo com a chave de assinatura correta, e o dispositivo aceitaria alegremente este firmware. Essa vulnerabilidade foi descoberta por Mike Davis da IOActive no início de 2014 e recebeu uma pontuação (CVSS) de 10.0 pela criticidade da vulnerabilidade.

Mais tarde, descobriu-se que o Belkin Wemo tinha vários outros problemas de segurança, incluindo bugs como injeção de SQL e modificação do nome do dispositivo para executar JavaScript arbitrário no smartphone Android do usuário, entre outros. Pesquisas adicionais foram realizadas no Belkin Wemo pelo grupo FireEye (consulte <https://www.fireeye.com/blog/threatresearch/2016/08/embedded_hardwareha.html>), que envolveu a obtenção de acesso ao firmware e console de depuração usando técnicas de hardware Universal Asynchronous Receiver Transmitter (UART) e Serial Peripheral Interface (SPI). Isso também os levou a identificar que, por meio do acesso ao hardware, alguém pode modificar facilmente os argumentos do bootloader, tornando inútil a verificação de assinatura do firmware do dispositivo.

**Bomba de insulina**

Um pesquisador de segurança chamado Jay Radcliffe, trabalhando para a Rapid7, identificou que alguns dispositivos médicos, especificamente bombas de insulina, poderiam estar sofrendo de uma vulnerabilidade a ataques baseados em replay. Radcliffe, ele próprio diabético tipo 1, decidiu pesquisar uma das bombas de insulina mais populares do mercado, o sistema de bomba de insulina OneTouch Ping da Animas, subsidiária da Johnson & Johnson. Durante a análise, ele descobriu que a bomba de insulina usava mensagens de texto claro para se comunicar, o que tornava extremamente simples para qualquer pessoa capturar a comunicação, modificar a dose de insulina a ser administrada e retransmitir o pacote. Quando ele testou o ataque na bomba de insulina OneTouch Ping, funcionou perfeitamente, sem nenhuma maneira de saber a quantidade de insulina que estava sendo administrada durante o ataque.

A vulnerabilidade foi corrigida pelo fornecedor, Animas, em cinco meses, o que mostra que pelo menos algumas empresas levam os relatórios de segurança a sério e tomam medidas para manter os clientes seguros.

**Fechaduras Inteligentes**

Um pesquisador de segurança com o apelido Jmaxx embarcou em um desafio para encontrar pontos fracos de segurança na fechadura inteligente August, considerada uma das fechaduras inteligentes mais populares e seguras, usada tanto por consumidores em suas casas quanto por anfitriões do Airbnb para permitir que os hóspedes façam o check-in quando for conveniente.

Algumas das vulnerabilidades que ele descobriu incluíam a capacidade dos hóspedes de se tornarem administradores modificando um valor no tráfego de rede de usuário para superusuário, firmware não assinado, funcionalidade do aplicativo para ignorar fixação de Secure Sockets Layer (SSL) (habilitando modo de debug) e muito mais.

No mesmo evento, os pesquisadores de segurança Anthony Rose e Ben Ramsey da empresa de segurança Merculite fizeram outra apresentação intitulada “Arrombando fechaduras Bluetooth Low Energy a 400 metros de distância”, na qual revelaram vulnerabilidades em uma série de produtos de fechaduras inteligentes, incluindo Quicklock Padlock, iBluLock Padlock, Plantraco Phantomlock, Ceomate Bluetooth Smart Doorlock, Elecycle EL797 e EL797G Smart Padlock, Vians Bluetooth Smart DoorlockOkidokey Smart Doorlock, Poly-Control Danalock Doorlock, Mesh Motion Bitlock Padlock e Lagute Sciener Smart Doorlock.

As vulnerabilidades descobertas por Rose e Ramsey eram de vários tipos, incluindo transmissão da senha em texto puro, suscetibilidade a ataques baseados em replay, engenharia reversa de aplicativos móveis para identificar informações confidenciais, fuzzing e spoofing de dispositivo. Por exemplo, durante o processo de redefinição de senha, o Quicklock Padlock envia um pacote Bluetooth Low Energy (BLE) contendo o opcode, a senha antiga e a nova senha. Como até mesmo a autenticação normal, acontece através de comunicação de texto puro, um invasor pode então usar a senha para definir uma nova senha para a fechadura da porta, tornando o dispositivo inútil para o proprietário original. A única maneira de reiniciá-lo seria remover a bateria do dispositivo após abrir o compartimento. Em outro dispositivo, o Danalock Doorlock, é possível fazer engenharia reversa do aplicativo móvel para identificar o método de criptografia e encontrar a chave de criptografia codificada ("thisisthesecret") usada.

**Hackeando armas e rifles inteligentes**

Além dos dispositivos e eletrodomésticos inteligentes típicos, os rifles também estão ficando inteligentes. A TrackingPoint, fabricante de tecnologia para rifles inteligentes, oferece um aplicativo móvel para visualizar e ajustar a mira do tiro. Este aplicativo foi considerado vulnerável a alguns problemas de segurança. Runa Sandvik e Michael Auger identificaram vulnerabilidades no rifle inteligente que lhes permitiram acessar interfaces de programação de aplicativos de administração (APIs) após obter acesso ao dispositivo via UART. Explorando o aplicativo móvel, um ataque baseado em rede permitiria a um invasor alterar vários parâmetros, como velocidade e direção do vento, peso da bala e outros parâmetros necessários para o disparo. Quando esses parâmetros são modificados, o atirador não saberia que essas alterações foram feitas.

Outro caso ocorreu quando um pesquisador de segurança conhecido como Plore conseguiu burlar algumas das restrições de segurança aplicadas pela IP1, uma arma inteligente da Armatix. A arma inteligente exigia que o atirador usasse um relógio especial fornecido pela IP1 para disparar a arma. Para contornar as restrições de segurança, Plore inicialmente realizou uma análise de sinal de rádio e encontrou a frequência exata que a arma usa para se comunicar. Mais tarde, ele percebeu que usando alguns ímãs, o pino de metal que trava o pino de disparo poderia ser manipulado, permitindo ao atirador disparar a bala. Mesmo que o uso de ímãs não seja um ataque de alta tecnologia que você possa pensar ser necessário para explorar dispositivos IoT, é um ótimo exemplo de como pensar fora da caixa pode ajudar a identificar vulnerabilidades.

Estas vulnerabilidades servem como exemplos para ajudá-lo a entender vários tipos de vulnerabilidades normalmente encontradas em dispositivos IoT. Mais tarde, abordaremos vários componentes de dispositivos IoT, incluindo técnicas para exploração de hardware, rádio, firmware e software. Você aprenderá mais sobre como usar algumas dessas técnicas nos dispositivos IoT que você está pesquisando ou realizando um teste de penetração.

**Fragmentação da Internet das coisas**

Como a IoT é um campo enorme, com toda empresa querendo sua fatia do bolo, você frequentemente encontrará vários protocolos e frameworks que podem ajudar os desenvolvedores a levar seus produtos ao mercado mais rapidamente.

Frameworks de IoT são várias ofertas disponíveis que ajudam os desenvolvedores de IoT a acelerar o processo de desenvolvimento de uma solução de dispositivo IoT, aproveitando a base de código e bibliotecas existentes oferecidas, reduzindo o tempo de lançamento do produto no mercado. Embora isso torne as coisas significativamente mais fáceis para desenvolvedores e empresas, o outro lado, que muitas vezes é negligenciado, é a segurança desses frameworks. Na verdade, com base em minhas experiências com testes de penetração em dispositivos IoT, dispositivos usando vários frameworks eram frequentemente vulneráveis ​​a problemas de segurança básicos. As discussões que tive posteriormente com as equipes de produto revelaram que a mentalidade geral é que, se alguém está usando um framework popular, ele geralmente é considerado seguro por design, resultando em descuido na avaliação de sua segurança.

Não importa de qual lado você esteja, os construtores ou os invasores, é importante observar os problemas de segurança do produto, independentemente da estrutura subjacente ou dos conjuntos de protocolos usados. Por exemplo, você frequentemente encontrará desenvolvedores usando ZigBee pensando que ele é extremamente seguro, deixando seus produtos vulneráveis ​​a todos os tipos de ataques baseados em rádio.

Neste livro, não nos concentramos necessariamente em nenhuma framework ou pilha de tecnologia específica, mas sim em uma abordagem aplicável a qualquer solução de dispositivo IoT, independentemente da arquitetura subjacente. Nesse processo, no entanto, também abordamos alguns protocolos populares (por exemplo, ZigBee e BLE) para lhe dar uma ideia de que tipo de vulnerabilidades esperar e como encontrar esses problemas de segurança.

Algumas das estruturas de IoT mais populares incluem o seguinte:

Eclipse Kura ( https://www.eclipse.org/kura/ )

The Physical Web ( https://google.github.io/physicalweb/ )

IBM Bluemix (now IBM Cloud: https://www.ibm.com/cloud/ )

Lelylan ( http://www.lelylan.com/ )

Thing Speak ( https://thingspeak.com/ )

Bug Labs ( https://buglabs.net/ )

The thing system ( http://thethingsystem.com/ )

Open Remote ( http://www.openremote.com/ )

OpenHAB ( https://www.openhab.org/ )

Eclipse IoT ( https://iot.eclipse.org/ )

Node-Red ( https://nodered.org/ )

Flogo ( https://www.flogo.io/ )

Kaa IoT ( https://www.kaaproject.org/ )

Macchina.io ( https://macchina.io/ )

Zetta ( http://www.zettajs.org/ )

GE Predix ( https://www.ge.com/digital/predixplatform-foundation-digital-industrialapplications )

DeviceHive ( https://devicehive.com/ )

Distributed Services Architecture ( http://iot-dsa.org/ )

Open Connectivity Foundation ( https://openconnectivity.org/ )

Isso é apenas uma pequena fração de alguns dos frameworks de dispositivos IoT mais populares que você encontrará ao mergulhar no mundo da IoT.

Da mesma forma, quando se trata de protocolos de comunicação, há toda uma gama de protocolos sendo usados pelos fabricantes para suas soluções de IoT.

Alguns dos protocolos de comunicação mais populares incluem o seguinte:

* Wi-Fi
* BLE
* Cellular/Long Term Evaluation (LTE)
* ZigBee
* ZWave
* 6LoWPAN
* LoRA
* CoAP
* SigFox
* Neul
* MQTT
* AMQP
* Thread
* LoRaWAN

Para avaliar adequadamente a segurança IoT de um determinado dispositivo ou protocolo de comunicação, você precisará de várias ferramentas de hardware. Por exemplo, o Ubertooth One seria necessário para capturar e analisar pacotes BLE, o Atmel RzRaven para ZigBee, e assim por diante.

Agora que temos uma boa ideia do que é IoT e das várias tecnologias envolvidas, vamos dar uma olhada em alguns dos fatores que levam à insegurança desses dispositivos.

**Razões para Vulnerabilidades de Segurança IoT**

Dado que os dispositivos IoT são extremamente complexos por natureza, é altamente provável que a maioria dos dispositivos que você encontrar terá problemas de segurança. Se tentarmos entender por que essas vulnerabilidades existem em primeiro lugar, e como você pode evitar esses problemas de segurança ao construir um produto, precisamos nos aprofundar em todo o ciclo de vida de desenvolvimento do produto, desde a fase de ideação até o produto ser lançado no mercado.

Alguns dos motivos que se destacam como causa de problemas de segurança ao construir esses dispositivos são apresentados a seguir.

**Falta de Conscientização de Segurança entre Desenvolvedores**

Desenvolvedores que trabalham nestes dispositivos inteligentes frequentemente possuem menos conhecimento, ou sequer conhecimento, sobre as possíveis vulnerabilidades de segurança em dispositivos IoT. Considerando que, em grandes organizações, os desenvolvedores geralmente já estão sobrecarregados, seria uma ótima ideia ter reuniões periódicas para discutir como eles podem construir produtos seguros desde o início, incluindo táticas práticas como diretrizes rígidas de codificação a serem seguidas e uma lista de verificação de segurança para qualquer amostra de código em que trabalhem.

**Falta de Perspectiva Macro**

Como veremos no próximo capítulo sobre os vários componentes que constituem um dispositivo de IoT, é extremamente fácil para desenvolvedores ou equipes de segurança esquecerem o fato de que é a interconexão de dispositivos e diversas tecnologias que pode levar a problemas de segurança. Por exemplo, apenas olhar para o aplicativo móvel pode não revelar problemas de segurança, mas se você combinar as descobertas do aplicativo móvel e como a comunicação de rede funciona, poderá descobrir uma falha crítica de segurança. É essencial que as equipes de produto invistam mais tempo e esforço analisando toda a arquitetura do dispositivo e realizando modelagem de ameaças.

**Problemas de segurança baseados na cadeia de suprimentos**

Uma das causas das vulnerabilidades de segurança em dispositivos IoT é o envolvimento de muitos participantes. Isso significa que você frequentemente encontrará diferentes componentes de dispositivos sendo fabricados por fornecedores diferentes, tudo sendo montado por outro fornecedor e, finalmente, sendo distribuído por outro ainda. Isso, embora inevitável na maioria das situações, pode levar a problemas de segurança (ou backdoor) que podem ser introduzidos por um deles, colocando todo o produto em risco.

**Utilização de Frameworks e Bibliotecas de Terceiros Inseguras**

No caso de dispositivos IoT ou qualquer outra tecnologia, é comum encontrar desenvolvedores usando bibliotecas e pacotes existentes, introduzindo amostras de código potencialmente vulneráveis em um produto seguro. Embora algumas organizações tenham verificações de qualidade no código escrito pelos desenvolvedores, muitas vezes elas tendem a negligenciar os pacotes que estão sendo utilizados. Isso também é acompanhado pelas exigências de negócios de uma organização, onde a gerência exige que os produtos cheguem ao mercado em prazos acelerados (geralmente irreais), o que coloca a avaliação de segurança do produto em segundo plano. Muitas vezes, sua importância não é percebida até que o produto sofra uma violação de segurança.

**Conclusão**

Neste capítulo, vimos o que são dispositivos IoT, os protocolos e frameworks usados por esses dispositivos inteligentes e os motivos pelos quais esses dispositivos geralmente são vulneráveis. Também examinamos alguns dos problemas de segurança identificados anteriormente em soluções populares de dispositivos IoT para entender quais são algumas das vulnerabilidades encontradas em dispositivos do mundo real. No próximo capítulo, examinaremos mais profundamente o mapeamento da superfície de ataque desses dispositivos e como podemos identificar e possivelmente evitar riscos de segurança em dispositivos IoT.

**2.Realizando um teste IoT de penetração**

Neste capítulo, aprenderemos como realizar um teste de penetração de IoT e entender o primeiro elemento dele, que é o mapeamento da superfície de ataque. Muitos tester’s(quem realiza os testes de penetração) ainda não conseguiram migrar para o teste de penetração de IoT devido à falta de conhecimento sobre como realizar um teste de penetração de IoT: Quais são os diferentes componentes envolvidos? Quais ferramentas devem ser usadas? Como você executa um teste geral?

Este capítulo compartilha insights sobre como realizar um teste de penetração de IoT e responder a essas perguntas. Também abordamos a primeira fase do processo de teste de penetração, mapeamento da superfície de ataque, que usamos para avaliar a solução do dispositivo IoT de destino e obter uma estimativa justa de quais tipos de problemas de segurança podem estar presentes no produto que estamos testando.

O que é um teste de penetração de IoT?

Um teste de penetração de IoT é a avaliação e exploração de vários componentes presentes em uma solução de dispositivo IoT para ajudar a tornar o dispositivo mais seguro. Ao contrário dos testes de penetração tradicionais, a IoT envolve vários componentes diferentes, como discutimos anteriormente, e sempre que falamos sobre um teste de penetração de IoT, todos esses componentes precisam ser testados.

Assim como em qualquer teste de penetração de IoT, nós, como profissionais da área, precisamos entender o escopo do teste, bem como quaisquer outras restrições e limitações. As condições do teste de penetração variam de produto para produto e podem incluir qualquer coisa, desde a garantia de que o teste ocorra entre 22h e 5h (ou durante a noite) até a realização do teste em um ambiente de “mentira” fornecido pelo cliente.

Uma vez que você compreenda o escopo técnico do projeto, vale a pena mencionar ao cliente qual tipo de teste de penetração (caixa branca, caixa preta ou caixa cinza) você ou sua equipe irá realizar para garantir que ambos estejam na mesma página.

Um dos outros pontos sobre o teste de penetração de IoT é a necessidade de vários dispositivos. Muitas vezes, durante um teste de penetração de IoT, certas técnicas que usamos envolvem métodos destrutivos, como a remoção de um chip de uma placa de circuito para análise, o que provavelmente tornaria o dispositivo inutilizável para análises posteriores.

Após as discussões, a próxima etapa é realizar o teste de penetração de acordo com o escopo e a metodologia desejados. Esta fase do teste de penetração começa com o mapeamento de toda a superfície de ataque da solução, seguido pela identificação de vulnerabilidades e realização de exploração, que é então seguida pela pós-exploração. O teste conclui com um relatório técnico aprofundado. Neste capítulo, abordaremos apenas a primeira etapa, o mapeamento da superfície de ataque. Nos próximos capítulos, veremos as várias maneiras de identificar e explorar vulnerabilidades e, no capítulo final, veremos como escrever um relatório de teste de penetração para dispositivos IoT.

**Mapeamento de superfície de ataque**

O processo de mapeamento da superfície de ataque significa mapear todos os diversos pontos de entrada que um invasor poderia potencialmente explorar em uma solução de dispositivo IoT. Este é o primeiro passo, e um dos mais importantes, em toda a metodologia de teste de penetração de IoT. Ele também envolve a criação de um diagrama de arquitetura de todo o produto da perspectiva do profissional de teste de penetração.

Durante os trabalhos de teste de penetração, frequentemente dedicamos um dia inteiro a esta fase. Esta etapa é útil porque ajuda a entender a arquitetura de toda a solução e, ao mesmo tempo, ajuda a estabelecer vários testes que você executaria no produto, classificados por prioridade. A prioridade dos ataques pode ser determinada pela facilidade de exploração multiplicada pelo impacto da exploração.

Em um caso em que a exploração é extremamente fácil e leva a um comprometimento bem-sucedido e recuperação de dados confidenciais do dispositivo, isso seria classificado como uma vulnerabilidade de alta prioridade e alta criticidade. Por outro lado, algo que é difícil de executar - com resultados obtidos durante o teste que não sejam tão úteis - seria categorizado como uma vulnerabilidade de baixa criticidade e baixa prioridade. Nos trabalhos de teste de penetração, sempre que identificamos uma vulnerabilidade de alta criticidade, também notificamos o fornecedor imediatamente sobre a visão geral da vulnerabilidade e seu impacto no mesmo dia, em vez de esperar a conclusão do trabalho.

Agora que você tem uma noção básica do que fazer no mapeamento da superfície de ataque, vamos nos aprofundar e entender os detalhes exatos de como realizar esse processo.

**Como realizar o mapeamento da superfície de ataque**

É crucial dedicar tempo para entender completamente o dispositivo antes de iniciar um teste de penetração. Realizar uma avaliação com informações incompletas ou parciais é um dos maiores erros que um profissional de teste de penetração pode cometer. Isso significa vasculhar todos os canais possíveis e coletar informações, como documentação e manuais do dispositivo, recursos online e publicações sobre o produto, e qualquer conteúdo disponível ou pesquisas anteriores sobre o dispositivo.

Anote os vários componentes usados no dispositivo, tipo de arquitetura da CPU, protocolos de comunicação usados, detalhes do aplicativo móvel, processo de atualização de firmware, portas de hardware, suporte a mídia externa nos dispositivos e praticamente qualquer outra coisa que você possa encontrar. Muitas vezes, as coisas não são tão óbvias quanto parecem inicialmente, e é por isso que você deve se aprofundar em cada uma das várias funções que o dispositivo oferece.

Ao analisar uma solução de IoT para mapeamento da superfície de ataque, podemos dividir toda a arquitetura em três categorias:

1. Dispositivo embarcado.
2. Firmware, software e aplicativos.
3. Comunicações de rádio.

Nosso objetivo ao analisar o dispositivo IoT para mapeamento da superfície de ataque é categorizar a funcionalidade e as ameaças de segurança correspondentes a cada categoria. Vamos considerar qual deve ser o processo de pensamento ao categorizar as vulnerabilidades potenciais de acordo com as categorias mencionadas. Cada uma das categorias mencionadas a seguir serve como uma introdução a esse componente e será detalhada com mais profundidade nos próximos capítulos.

**Dispositivos incorporados**

Um dispositivo embarcado é a peça central de qualquer arquitetura de dispositivo IoT e também é o "objeto" na Internet das Coisas. O dispositivo embarcado em um produto IoT pode ser usado para diversas finalidades, dependendo do cenário de uso. Ele pode funcionar como um hub para toda a arquitetura IoT do dispositivo, como sensor que coleta dados do ambiente físico ao seu redor, ou como forma de exibir dados ou realizar a ação desejada pelo usuário. Assim, os objetos na Internet das Coisas podem ser usados para coletar, monitorar, analisar dados e realizar ações.

Para esclarecer isso com um exemplo do mundo real, pense em um produto IoT para casas inteligentes. Existem muitos dispositivos que juntos formam o produto IoT para casa inteligente. Isso inclui uma porta de entra inteligente ou ponto central, lâmpadas inteligentes, sensores de movimento, interruptores inteligentes e dispositivos conectados adicionais.

Mesmo que os dispositivos tenham finalidades diferentes, na maioria das vezes, a abordagem para testar a segurança desses dispositivos contra vulnerabilidades seria a mesma. Dependendo da finalidade do dispositivo, ele conterá informações confidenciais que, se comprometidas, seriam consideradas críticas.

A seguir estão algumas das vulnerabilidades encontradas em dispositivos embarcados:

* Portas seriais expostas.
* Mecanismo de autenticação inseguro usado nas portas seriais.
* Capacidade de despejar o firmware via JTAG ou chips Flash.
* Ataques baseados em mídia externa.
* Análise de energia e ataques baseados em canais laterais.

Para avaliar a segurança do dispositivo, o processo de reflexão deve ser baseado nessas perguntas: Quais são as funcionalidades do dispositivo? A quais informações o dispositivo tem acesso? Com base nesses dois fatores, podemos estimar realisticamente os potenciais problemas de segurança e seu impacto.

Uma vez que nos aprofundemos na exploração de hardware, no Capítulo 3, entenderemos melhor as falhas subjacentes em dispositivos IoT comuns e veremos como podemos explorar as várias vulnerabilidades de segurança de hardware que encontramos em dispositivos IoT.

**Firmware, Software e Aplicativos**

Após a exploração do hardware, o próximo componente que analisamos é a parte de software de um dispositivo IoT. Isso inclui tudo, desde o firmware que roda no dispositivo, os aplicativos móveis usados para controlá-lo, os componentes de nuvem conectados a ele e assim por diante.

São também nesses componentes que você pode aplicar a experiência tradicional de pentest ao ecossistema de IoT. Isso também envolveria tópicos como engenharia reversa de binários de diferentes arquiteturas, incluindo Advanced RISC Machines (ARM) e MIPS, bem como engenharia reversa de aplicativos móveis. Esses componentes podem frequentemente ajudá-lo a descobrir muitos segredos e encontrar vulnerabilidades. Dependendo do componente que você está testando, utilizará diferentes conjuntos de ferramentas e técnicas variadas.

Um dos outros objetivos durante o pentest de componentes baseados em software é analisar as várias maneiras pelas quais podemos acessar o componente individual que queremos testar. Por exemplo, se quisermos analisar o firmware em busca de vulnerabilidades, precisaríamos ter acesso ao firmware, o que geralmente não é fácil.

Também precisamos concentrar muitos esforços na engenharia reversa das APIs de comunicação que nos ajudam a entender como os diferentes componentes do dispositivo IoT interagem entre si e a verificar quais tipos de protocolos de comunicação estão em uso.

Se olharmos para um dispositivo IoT do mundo real, uma casa inteligente terá os seguintes componentes que serão abordados na seção de software:

## Aplicativo móvel: permite controlar dispositivos inteligentes - ligando e desligando as luzes, adicionando novos dispositivos ao sistema de casa inteligente e assim por diante. Normalmente, você terá aplicativos móveis para as plataformas Android e iOS, que são as duas plataformas de aplicativos móveis dominantes atualmente, esse texto. Existem vários ataques possíveis em aplicativos móveis que podem revelar informações confidenciais do dispositivo ou como ele funciona. Eles também podem servir como um ponto de entrada para atacar o componente web (mencionado posteriormente) por meio da engenharia reversa do binário do aplicativo e de suas APIs de comunicação. Em relação aos aplicativos móveis, também podemos precisar trabalhar com componentes nativos do aplicativo, o que pode nos levar a uma compreensão adicional de todo o binário do aplicativo e várias funcionalidades subjacentes, como criptografia e outros aspectos sensíveis.

Painel baseado na web: Isso permite ao usuário monitorar o dispositivo, visualizar análises e informações de uso, controlar permissões para os dispositivos e assim por diante. A maioria dos dispositivos IoT que você encontrará terá uma interface web onde você pode acessar os dados enviados do dispositivo para o terminal web. Se o aplicativo web for vulnerável, ele pode permitir que você acesse dados não autorizados, que podem ser os dados do mesmo usuário ou de qualquer outro usuário usando o mesmo dispositivo IoT, o que já aconteceu com muitos dispositivos IoT no passado, principalmente monitores de bebês.

Interfaces de rede inseguras são componentes de dispositivos IoT expostos à rede e que podem ser comprometidos devido a vulnerabilidades. Isso pode acontecer de duas formas: por meio de uma porta aberta que aceita conexões a serviços sem qualquer tipo de autenticação, ou por um serviço que esteja executando uma versão vulnerável e desatualizada, com falhas de segurança conhecidas para aquela versão específica.

Já realizamos testes de penetração em diversos dispositivos que executavam versões vulneráveis de componentes como o Simple Network Management Protocol (SNMP - Protocolo Simples de Gerenciamento de Rede) e o File Transfer Protocol (FTP - Protocolo de Transferência de Arquivos).

## Firmware: O Coração do Dispositivo

O firmware controla os vários componentes do dispositivo e é responsável por todas as suas ações. Pense nele como o componente que detém as chaves do reino. Praticamente tudo o que você possa imaginar ser extraído do dispositivo pode ser encontrado no firmware. O capítulo dedicado ao firmware neste livro explica o que é firmware, seu funcionamento interno, as várias vulnerabilidades que podemos encontrar e como realizar análises adicionais.

## Comunicação entre Dispositivos IoT

Aplicativos móveis, aplicativos web e dispositivos embarcados costumam se comunicar com outros componentes e terminais back-end por meio de diferentes mecanismos de comunicação, como o Representational State Transfer (REST - Transferência de Estado Representacional). Transferência (REST), Simple Object Access Protocol (SOAP- Protocolo de Acesso Simples a Objetos), Message Queuing Telemetry Transport (MQTT- Transporte de telemetria de enfileiramento de mensagens), Constrained Application Protocol (CoAP- Protocolo de aplicação restrito) e muito mais, que abordaremos brevemente nos próximos capítulos.

Além disso, alguns componentes coletariam dados e os enviariam para um ponto final remoto com frequência, o que muitas vezes poderia ser tratado como uma violação de privacidade, mais apropriadamente, do que um problema de segurança. Todo o foco do mapeamento da superfície de ataque é garantir que você tenha informações suficientes para entender todos os aspectos e funcionalidades do dispositivo, o que nos ajudará a compreender os problemas de segurança neles.

Esses componentes envolvem muitas vulnerabilidades, algumas das quais estão listadas aqui.

**Firmware**

* Firmware
* Capacidade de modificar o firmware
* Verificação insegura de assinatura e integridade
* Valores confidenciais codificados no firmware - chaves de API, senhas, URLs de “mentira” e assim por diante
* Certificados privados
* Capacidade de entender toda a funcionalidade do dispositivo através do firmware
* Extração do sistema de arquivos do firmware
* Componentes desatualizados com vulnerabilidades conhecidas

**Aplicativos móveis**

* Engenharia reversa do aplicativo móvel
* Extração do código-fonte do aplicativo móvel
* Verificações inseguras de autenticação e autorização
* Falhas de lógica e negócio
* Vazamento de dados por canal lateral
* Ataques de manipulação de “runtime”
* Comunicação de rede insegura
* Bibliotecas de terceiros e kits de desenvolvimento de software (SDKs) desatualizados

**Aplicativo web**

* Injeção do lado do cliente
* Referência insegura de objeto direto
* Autenticação e autorização inseguras
* Vazamento de dados confidenciais
* Falhas na lógica de negócios
* Falsificação de requisição entre sites (CSRF)
* “Scripting” entre sites (XSS)

Essa lista é apenas um exemplo de algumas das vulnerabilidades presentes nesses componentes, o que deve lhe dar uma ideia do tipo de vulnerabilidade que afeta esses componentes.

**Comunicações de Rádio**

As comunicações de rádio fornecem uma maneira para diferentes dispositivos se comunicarem entre si. Essas mídias e protocolos de comunicação geralmente não são considerados pelas empresas ao pensar em segurança, tornando-se, portanto, um ponto ideal para que os testadores de penetração identifiquem vulnerabilidades em dispositivos IoT.

Alguns dos protocolos comuns de comunicação por rádio usados em dispositivos IoT são celular, Wi-Fi, BLE, ZigBee, Wave, 6LoWPAN, LoRa e outros. Dependendo do protocolo de comunicação que um dispositivo está usando, hardware especializado pode ser necessário para realizar a análise da comunicação por rádio.

Durante o processo de análise inicial, você também deve listar todos os diferentes itens de hardware e software necessários para realizar uma avaliação de segurança dos protocolos de rádio em uso. Embora inicialmente possa parecer uma tarefa onerosa, uma vez que você tenha adquirido as ferramentas necessárias para realizar a avaliação, é apenas uma questão de analisar a comunicação usando essas ferramentas.

Configurar software e ferramentas para teste de penetração de rádio (e outros componentes de teste de penetração de IoT) pode ser uma tarefa árdua. É por isso que construímos uma máquina virtual (VM) personalizada chamada AttifyOS que você pode usar para todos os exercícios e laboratórios de teste de peneteação de IoT abordados neste livro. Você pode baixar o AttifyOS em <https://www.attify.com/attifyos>.

Ao longo deste livro, abordamos três categorias principais em comunicação por rádio que são mais relevantes do ponto de vista de teste de penetração e avaliação de segurança:

Rádio Definido por Software (SDR).

Exploração de ZigBee.

Exploração de BLE (Bluetooth Low Energy).

Dependendo do componente de rádio com o qual estamos trabalhando, ele terá diferentes conjuntos de vulnerabilidades. No entanto, estes são os tipos mais comuns de vulnerabilidades que encontramos em protocolos e meios de comunicação por rádio:

* Ataques Man-in-the-middle.
* Ataques baseados em replay.
* Verificação insegura de Cyclic Redundancy Check (CRC).
* Ataques baseados em jamming.
* Negação de serviço (DoS).
* Falta de criptografia.
* Capacidade de extrair informações confidenciais de pacotes de rádio.
* Interceptação e modificação de comunicação de rádio ao vivo.

Abordaremos essas categorias de ataque e maneiras de realizá-las nos capítulos posteriores deste livro. Ao criar um mapa da superfície de ataque para comunicação por rádio, o processo deve se concentrar nos seguintes itens:

Quais são as funções dos vários componentes envolvidos?

Qual componente inicia o mecanismo de autenticação e emparelhamento?

Como é a aparência do mecanismo de emparelhamento?

Quantos dispositivos cada componente pode manipular simultaneamente?

Em qual frequência o dispositivo opera?

Quais protocolos estão sendo usados por diferentes componentes? Eles são protocolos personalizados ou proprietários?

Existem dispositivos semelhantes operando na mesma faixa de frequência deste dispositivo?

Esses são apenas alguns dos itens que você deve considerar ao analisar a comunicação de rádio para um determinado dispositivo IoT.

Criando um Mapa da Superfície de Ataque

Agora que estamos familiarizados com todos os diferentes componentes que analisaremos e os tipos de vulnerabilidades que afetam os componentes, estamos em uma boa posição para criar um mapa da superfície de ataque de qualquer dispositivo IoT. A Figura 2-2 mostra o processo para criar um mapa da superfície de ataque.

Os passos a seguir descrevem como criar um mapa da superfície de ataque para qualquer dispositivo IoT:

1. Listar todos os componentes presentes no produto alvo.
2. Preparar um diagrama de arquitetura.
3. Rotular os componentes e os fluxos de comunicação entre eles.
4. Identificar vetores de ataque para cada componente e o canal ou protocolo de comunicação utilizado.
5. Categorizar os vetores de ataque com base na criticidade.

O diagrama de arquitetura inicial também nos auxilia durante todo o processo de compreensão da solução IoT e dos diversos componentes envolvidos. Certifique-se de listar todos os componentes envolvidos, por menores que pareçam, juntamente com todas as especificações técnicas de cada componente, durante o processo de criação do diagrama de arquitetura.

Para algumas informações que podem ser difíceis de obter inicialmente, como a frequência em que o dispositivo opera, é possível encontrar dados disponíveis online. Comece por sites como o fccid.io, onde você pode inserir o ID FCC de um dispositivo IoT e encontrar diversas informações sobre ele.

Por exemplo, vamos pegar o kit “Samsung Smart Things”, que consiste em vários dispositivos para automação residencial inteligente. Através de uma análise inicial do site, podemos descobrir que ele contém os seguintes itens:

* Central de Casa Inteligente
* Sensor de movimento
* Tomada inteligente
* Sensor de presença
* Sensor de movimento

Além disso, também possui um aplicativo móvel disponível na Google Play Store e Apple AppStore. O próximo passo é desenhar um diagrama desses componentes para nos ajudar a visualizá-los melhor. A Figura 2-3 é um exemplo de diagrama de arquitetura que criei para um dispositivo doméstico inteligente de amostra.

Os seguintes componentes estão envolvidos neste sistema de casa inteligente:

* Dispositivos.
* Aplicativo móvel.
* Gateway IoT.
* Recursos de nuvem.
* Protocolos de comunicação: BLE, Wi-Fi, ZigBee, ZWave, 6LoWPAN, GSM e Ethernet.
* Os dispositivos e o aplicativo móvel se comunicam via BLE.
* O hub inteligente e os dispositivos se comunicam por meio de vários protocolos, como ZigBee, ZWave e 6LoWPAN.
* O aplicativo móvel também pode interagir com o hub inteligente via Wi-Fi.
* O aplicativo móvel e o hub inteligente se comunicam com a nuvem a cada cinco minutos e compartilham dados.
* O aplicativo móvel usa uma API REST para se comunicar por meio dos recursos de nuvem.
* Podemos ver esses detalhes adicionais especificados na Figura 2-3:
  + O gateway do hub inteligente possui uma porta Ethernet e um slot para cartão SD externo que pode ser usado para atualização de firmware.
  + O dispositivo contém um processador Broadcom.
  + O aplicativo móvel é um aplicativo nativo com a possibilidade de possuir bibliotecas adicionais.
* Durante o processo de configuração inicial, o dispositivo é configurado com uma senha padrão de "admin".
* O aplicativo Android ainda funciona se houver um problema de certificado; ou seja, o aplicativo funciona em conexões inseguras com autoridades certificadoras (CAs) não confiáveis para o certificado SSL.

Como você pode ver na Figura 2-3, temos todos os diferentes componentes mencionados no diagrama, juntamente com os vários canais de comunicação e protocolos que os vários dispositivos usam para se comunicar entre si ou com os terminais da web.

Depois de examinar este diagrama e todas as especificações técnicas, quando iniciamos nosso teste de penetração, agora sabemos exatamente como abordar esses dispositivos e quais são nossos alvos detalhados. Nesta etapa, precisamos pensar como um atacante. Se você tivesse que atacar um componente, como faria isso?

Quais vulnerabilidades você procuraria? Quais casos de teste você realizaria? Explorar esse componente específico é em que você deve se concentrar.

Com base em todas essas informações, prepare uma planilha com todos os casos de teste e explorações a serem testados nos diversos componentes, incluindo uma descrição detalhada de qual teste específico você realizará e qual será a saída se o ataque for bem-sucedido. Quanto mais detalhada for a sua planilha, mais eficaz será o seu teste de penetração. Se você estiver trabalhando em equipe, esta planilha é algo que você deve debater com sua equipe e depois ajustar. A Figura 2-4 mostra uma planilha de amostra.

Você também pode aproveitar os recursos disponíveis em vários lugares, incluindo:

* Guia de teste de penetração de IoT da Attify disponível em <http://www.iotpentestingguide.com> (esse link pode ser removido se você não quiser incluir o endereço).
* Guia de Hacking Embarcado pela OWASP.
* Superfície de Ataque IoT da OWASP.

**Estruturação do teste de penetração**

Como o teste de penetração de IoT é relativamente novo em comparação com outras formas de teste de penetração, poucas pessoas estão familiarizadas com a forma de executar o teste de penetração completo. Esta seção explica como estruturar o teste de penetração, o tamanho ideal da equipe, o número de dias necessários e outros detalhes relevantes.

Novamente, tudo isso vem da experiência pessoal de testes de penetração em centenas de dispositivos IoT nos últimos anos - e de encontrar problemas críticos de segurança em quase todos eles. Acredito que essa abordagem funcione de forma eficiente. Se você tiver outra abordagem para executar o teste de penetração que funcione melhor para você, certamente poderá continuar com ela.

Em seguida, a estrutura geral de um teste de penetração de IoT é explicada em detalhes.

**Engajamento do Cliente e Ligação Inicial para Discussão**

Esta é a ligação inicial para discussão após recebermos uma solicitação de uma organização para realizar um teste de penetração em seu dispositivo IoT. Mesmo antes desta etapa, temos uma discussão inicial com nossa equipe técnica para verificar se temos experiência relevante para o dispositivo IoT em questão e outros requisitos logísticos - recursos disponíveis, próximas datas disponíveis e assim por diante.

Durante esta etapa, colocamos nosso líder de teste de penetração em uma ligação com o cliente para discutir o dispositivo. Aqui estão algumas das perguntas que abordamos: Qual é o resultado esperado do teste de penetração? Quais componentes eles querem focar mais? Eles gostariam de um teste normal ou um teste de penetração com uma equipe de pesquisa adicional envolvida?

Se você é um tester’s(profissional que realiza testes), não posso enfatizar demais que seus clientes são seus ativos mais valiosos; é extremamente importante que você forneça serviços e ofertas apenas no domínio em que você e sua equipe se destacam. Dessa forma, você poderá atender melhor o cliente e construir um relacionamento duradouro.

**Discussão técnica adicional e chamada informativa**

Uma vez que decidimos que este é o projeto em que queremos trabalhar, e seríamos capazes de agregar valor ao engajamento geral com uma ótima pesquisa, pedimos ao cliente que sua equipe técnica participe de uma discussão com nossa equipe de teste de penetração que trabalhará nesse engajamento. Lembre-se, esta etapa ocorre após a assinatura de um acordo de confidencialidade e outra documentação necessária para que o cliente possa compartilhar livremente as especificações técnicas do produto.

Fazemos muitas perguntas durante esta fase para entender melhor o produto. Isso nos permite entender melhor o produto e explicar ao cliente nossa metodologia de teste de penetração e o que eles podem esperar durante cada etapa do teste de penetração. Também compartilhamos nosso mecanismo de relatório seguro, sistema de relatório diário, casos de teste que realizaremos, equipe de avaliação, interagindo com seu mecanismo de back-end e assim por diante. É importante ser transparente e justo com o cliente em relação à sua metodologia de teste de penetração e aos resultados que eles devem esperar a cada dia, ao final de cada fase e ao final do engajamento.

Também precisamos entender o processo de desenvolvimento deles, que tipo de teste sua equipe de segurança executa, se o teste de garantia de qualidade (QA) envolve testes de segurança, se existe um ciclo de vida de desenvolvimento seguro e assim por diante. Isso também ajuda a apresentar as equipes umas às outras, pois também oferecemos suporte personalizado aos desenvolvedores quando eles estão corrigindo as vulnerabilidades.

Obviamente, a maioria desses componentes corresponde a uma avaliação de caixa cinza, mas você entende a ideia. Durante um teste de penetração de caixa preta, você omitiria detalhes que um invasor não teria, o que também é chamado de simulação de exploração de atacante. Uma simulação de exploração de atacante é um método de teste de penetração no qual você compromete e ataca o dispositivo final da maneira que um atacante altamente direcionado faria.

**Explotação Simulada de Ataque**

Esta é a fase real de teste de penetração onde encontramos vulnerabilidades em produtos de IoT e as exploramos. Uma vez que recebemos os dispositivos em nossos laboratórios, nosso processo de teste de penetração é executado paralelamente com diversas atividades acontecendo ao mesmo tempo: nossa equipe de engenharia reversa trabalha na engenharia reversa de vários binários, a equipe de hacking embarcado invade o dispositivo de hardware IoT, a equipe de Rádio Definido por Software (SDR) trabalha na exploração da comunicação via rádio e a equipe de teste de penetração de software trabalha em firmware, aplicativos móveis, aplicativos web e ativos baseados em nuvem.

Isso só é possível se você tiver uma equipe forte, com diferentes divisões de teste de penetração e indivíduos com expertise em sua área de atuação. Se você é um pesquisador de segurança individual, também pode fazer isso, mas para teste de penetração de IoT, eu recomendo fortemente construir uma equipe de pelo menos três pessoas - especialistas em software e firmware, hardware e rádio - antes de realizar testes de penetração.

Uma vez que o engajamento é concluído, compartilhamos um relatório altamente detalhado junto com scripts de PoC, demonstrações em vídeo de alta qualidade, técnicas usadas para encontrar as vulnerabilidades, etapas para reprodução, métodos de correção e referências adicionais que fornecem mais informações sobre as vulnerabilidades identificadas.

## **Remediação**

Após concluirmos o teste de penetração, trabalhamos com os desenvolvedores, oferecendo suporte por voz, videochamada e e-mail, identificando exatamente o que precisa ser mudado e quais patches precisam ser implementados. Embora todas essas informações sejam fornecidas no relatório técnico, descobrimos que trabalhar com os desenvolvedores durante esta fase e oferecer suporte os ajuda a corrigir bugs com mais rapidez e a evitar cometer os mesmos erros novamente, graças ao aprendizado com nossa equipe durante as discussões de remediação.

## **Revalidação**

Uma vez que as vulnerabilidades de segurança tenham sido corrigidas pelos desenvolvedores, realizamos outro teste de penetração para as vulnerabilidades identificadas no teste inicial. Isso garante que todos os patches estejam instalados e que os patches aplicados pelos desenvolvedores sejam seguros e não causem vulnerabilidades em outros componentes. Esse é um dos erros que vemos os testadores de penetração cometerem: uma vez que o dispositivo é corrigido, eles limitam o teste de revalidação apenas aos componentes que consideraram vulneráveis. No entanto, é preciso prestar muita atenção para garantir que a correção do código naquele local não tenha levado à criação de bugs em outro. É assim que concluímos nosso teste de penetração para aquela versão do dispositivo.

## **Conclusão**

Neste capítulo, aprendemos como iniciar um teste de penetração de IoT, criando um modelo de ameaça, também conhecido como mapeamento da superfície de ataque para o produto. Também exploramos o que há por trás da superfície e demos uma olhada nos vários componentes presentes em uma arquitetura de IoT e as vulnerabilidades de segurança que poderíamos encontrar nesses componentes.

## **Ponto de ação**

1. Pegue qualquer dispositivo IoT ao seu redor (ou pense em um) e crie um diagrama da arquitetura desse dispositivo.
2. Depois de criar o diagrama da arquitetura, adicione os detalhes de como os dispositivos interagem entre si: quais componentes se conectam a quais e qual meio de comunicação e protocolo estão sendo usados.
3. Liste os problemas de segurança correspondentes a cada nó e a cada meio no diagrama que você criou.

Para receber feedback sobre sua criação e processo de pensamento, envie-me um e-mail com uma foto do seu diagrama e quaisquer anotações adicionais para:

[iothandbook@attify.com](mailto:iothandbook@attify.com)

**3.Anasilando Hardware**

Este é provavelmente o capítulo mais importante para você se nunca mexeu com hardware antes. Neste capítulo, veremos como podemos entender o hardware de um dispositivo IoT do ponto de vista da segurança, tanto para análise interna quanto externa. O dispositivo, como vimos nos capítulos anteriores, é um dos principais componentes em qualquer produto IoT. É o componente do dispositivo que pode ajudar a revelar muitos segredos sobre o dispositivo para nós, o que também veremos mais adiante neste capítulo.

A realização de análises de hardware pode ajudá-lo com as seguintes tarefas:

* Extrair firmware do dispositivo IoT do mundo real.
* Ganhar “root shell” no dispositivo para obter acesso irrestrito.
* Executar depuração ao vivo para ignorar proteções e restrições de segurança.
* Gravar novo firmware no dispositivo.
* Estender a funcionalidade do dispositivo.

Em alguns casos, abrir um dispositivo pode fazer com que ele não funcione corretamente (devido a violação física) ou você não conseguir remontá-lo. É por isso que, sempre que você estiver realizando um teste de penetração de dispositivo IoT, deve sempre pedir dois (ou mais) conjuntos de dispositivos ao cliente para que possa realizar avaliações de segurança física em um deles e o restante dos testes de vulnerabilidade no outro.

Se você nunca abriu hardware antes, tome cuidado especial ao trabalhar com os procedimentos deste capítulo para não se machucar. Seja sempre gentil e descubra uma maneira de abrir o dispositivo com cuidado para que possa continuar usando-o posteriormente, você pode colocá-lo de volta inteiro depois da análise. Agora, vamos começar.

**Inspeção Externa**

O primeiro passo na análise física do dispositivo é realizar uma inspeção externa. Isso inclui fazer um exame básico do dispositivo observando seus vários aspectos, incluindo itens como:

* Quantos e quais botões estão presentes.
* Opções de interface externa: porta Ethernet, slot para cartão SD e muito mais.
* Qual tipo de display o dispositivo possui.
* Requisitos de energia e voltagem para o dispositivo.
* Se o dispositivo possui alguma certificação e o que elas significam.
* Quais etiquetas de identificação FCC estão na parte traseira.
* Que tipo de parafusos o dispositivo usa.
* Se o dispositivo se parece com outros dispositivos com funcionalidades semelhantes que você já viu no mercado (talvez seja apenas um modelo renomeado).
* E assim por diante (você entende a ideia!).

Esta análise inicial lhe dará uma melhor compreensão do dispositivo como um todo e de como ele funciona, ao mesmo tempo que o ajudará a entender alguns detalhes internos do dispositivo.

Antes mesmo de abrir o dispositivo, há algumas coisas que você pode fazer apenas realizando esta análise inicial. A análise inicial normalmente envolve uma inspeção visual do dispositivo e uma revisão de outras fontes de informação sobre ele. Esta etapa também envolve usar o dispositivo e descobrir qual é sua funcionalidade normal. Depois de determinar a funcionalidade normal do dispositivo, você poderá definir abordagens direcionadas para subverter sua funcionalidade.

**Trabalhando com um Dispositivo Real**

Vamos pegar um dispositivo de amostra e começar a observá-lo conforme descrito. Neste caso, o dispositivo é um sistema de navegação e o modelo é Navman N40i. Apenas com uma rápida pesquisa inicial no Google, você pode aprender várias especificações do dispositivo, como esses:

1. Ele roda o Windows CE 5.0.
2. Possui uma câmera de 1.3 MP.
3. Oferece cinco horas de bateria.
4. Possui um processador Samsung 2400 de 400 MHz.
5. Possui 64 MB de SDRAM.
6. Possui 256 MB de ROM.
7. Contém um chip GPS SiRF STAR II. Estas informações úteis serão valiosas se mais tarde decidirmos encontrar vulnerabilidades no sistema Navman. Este rápido exemplo ilustra como abordar seu dispositivo assim que o obtiver e a análise inicial que você deve realizar.

Identificando Portas de Entrada e Saída O próximo passo é entender como funciona a entrada e saída (I/O) do dispositivo e o número de portas de E/S e outras conexões. Na Figura 3-1, podemos ver que o sistema Navman consiste em uma tela de 3,5 polegadas com cinco botões na frente, juntamente com um LED indicador na esquerda.

É assim que realizamos uma inspeção visual externa de um determinado dispositivo IoT. Lembre-se, esta é apenas a primeira etapa da análise do hardware. Para realizar uma boa inspeção, você precisa analisar os componentes externos e internos, juntamente com a criação de um mapa da superfície de ataque, conforme discutido no Capítulo 2.

**Inspeção Interna**

Após a inspeção externa, passamos para a inspeção interna. Como o próprio nome sugere, isso envolve abrir o dispositivo e observar seus componentes internos para melhor compreendê-lo e identificar as possíveis superfícies de ataque.

Para abrir o dispositivo, precisamos desapertar os parafusos. Os dispositivos IoT podem ter vários tipos de parafusos e, frequentemente, um conjunto de chaves de fenda comum não consegue abrir alguns dos tipos menos comuns encontrados nos dispositivos. Certifique-se de ter um bom conjunto de chaves de fenda à mão sempre que realizar exploração de IoT no dispositivo. Além disso, tome cuidado especial ao abrir o dispositivo para não danificar e seus circuitos internos. Um dos erros mais comuns que vejo as pessoas cometerem é tentar forçar a abertura do dispositivo, o que geralmente leva a danos físicos ao dispositivo, em alguns casos tornando-o não funcional.

Se você estiver vendo o interior do dispositivo pela primeira vez na vida, pode ficar fascinado com o que verá. O interior do dispositivo geralmente envolve muitos componentes, incluindo a placa de circuito impresso (PCB), conectores, antena, periféricos e assim por diante. Tente abrir o dispositivo alvo com cuidado e remover todos os cabos, fitas ou quaisquer outros periféricos conectados, um por um (veja a Figura 3-6).

Se você observar atentamente a Figura 3-6, poderá ver um módulo de câmera, uma bateria, um fone de ouvido, um conector GPS e uma antena GPS na parte superior com um LCD conectado por um cabo flexível. Se você olhar para o outro lado da placa, a Figura 3-7 mostra o que você verá.

Agora vamos analisar os diferentes componentes e suas funcionalidades, começando pelo processador. A Figura 3-8 mostra o processador usado em nosso sistema de navegação.

O processador é um dos componentes vitais de qualquer dispositivo IoT. Neste caso, o processador utilizado é o S3C2440AL, que é um processador Samsung ARM. Se pesquisarmos online por S3C2440AL, poderemos encontrar a folha de dados do processador, o que nos levaria a informações mais detalhadas sobre ele (Figura 3-9). A folha de dados deste processador pode ser encontrada em <https://www.keil.com/dd/docs/datashts/samsung/s3c2440_um.pdf>. Ele contém informações como portas de E/S, interrupções, Relógio de Tempo Real (RTC), Interface Serial Periférica (SPI) e muito mais.

Em seguida, podemos examinar outros componentes, como SDRAM e ROM, que estão presentes no dispositivo, conforme mostrado na Figura 3-10.

Na Figura 3-10, os componentes usados possuem os números K4M561633G, o que, pesquisando online, podemos ver que é uma SDRAM Móvel de 4M × 16 bits × 4 Bancos da Future Electronics, e também podemos ver 512 MB de ROM nele. Continuando, podemos continuar procurando por diferentes componentes - identificando seus números de peça e depois consultando-os online para aprender mais sobre eles.

Uma outra maneira de identificar componentes é observando seus logotipos e consultando um catálogo de referência online, como <https://www.westfloridacomponents.com/>.

Para procurar folhas de dados, você pode simplesmente pesquisar online pelo número do componente ou visitar um dos sites que contêm catálogos de folhas de dados, como <https://www.alldatasheet.com/> ou <https://www.datasheets360.com/>.

Há um último ponto que ainda não vimos, que provavelmente é o aspecto mais importante: as portas e interfaces de debug. Frequentemente, os dispositivos expõem interfaces de comunicação que podem ser exploradas para obter acesso adicional ao dispositivo para realizar ações como ler os logs de debug ou até mesmo obter um shell root não autenticado no dispositivo alvo. Como você pode ver na Figura 3-11, em nosso dispositivo, temos o dispositivo expondo interfaces que poderíamos usar para nos comunicar com o dispositivo usando UART e JTAG.

Essas interfaces podem ser encontradas apenas olhando para a PCB e identificando Tx e Rx para UART e TRST, TMS, TDO, TDI e TCK para JTAG, ambos os quais abordaremos detalhadamente nos próximos capítulos. Se você não estiver familiarizado com esses termos, não se preocupe, pois é aqui que passaremos a maior parte do nosso tempo de hacking de hardware no restante do livro.

Analisando Folhas de Dados Muitos dispositivos podem não ter muitas informações técnicas disponíveis em seu site oficial. É aqui que o banco de dados de ID da FCC vem em socorro. Se você é um engenheiro eletrônico e deseja se aprofundar no dispositivo e talvez até olhar para os esquemas do dispositivo, onde você iria? O banco de dados da FCC é a resposta. Então, o que é o banco de dados da FCC, você pode perguntar.

## **O que é o ID FCC?**

A Comissão Federal de Comunicações (FCC) é um órgão regulador responsável por diversos dispositivos que emitem sinais de rádio (a maioria dos dispositivos de IoT se encaixa nessa categoria). A regulamentação existe porque o espectro de rádio é limitado e há aparelhos operando em frequências diferentes.

Sem um órgão regulador, um fabricante poderia escolher qualquer frequência para o seu dispositivo, mesmo que já estivesse sendo usada por outro, causando interferência na comunicação de outros equipamentos.

Portanto, qualquer dispositivo que utilize comunicação via rádio precisa passar por um processo de aprovação, que envolve diversos testes. Após a aprovação, a FCC concede um código de identificação exclusivo para o dispositivo. O ID FCC é o mesmo para um determinado modelo de um fabricante específico. É importante ressaltar que o ID FCC não é uma permissão para transmissão, mas sim a aprovação de uma agência reguladora do governo dos Estados Unidos.

Você pode encontrar o ID FCC do dispositivo impresso nele ou pesquisando online em diferentes fontes. Não confunda o ID FCC com dispositivos que apenas cumprem as regulamentações da FCC, pois eles podem não precisar de um ID FCC, já que não se comunicam sem fio e geram apenas pequenas quantidades de ruído de rádio não intencional.

As informações sobre o processo de teste estão disponíveis no site da FCC, a menos que o fabricante solicite confidencialidade do documento. Você pode pesquisar informações sobre um dispositivo usando o ID FCC no site oficial da FCC em <https://www.fcc.gov/oet/ea/fccid> ou em sites não-oficiais como fccid.io ou fcc.io.

## **Usando o ID FCC para encontrar informações sobre o dispositivo**

Vamos pegar um dispositivo comercial real e usar o ID FCC para encontrar informações sobre ele. Nesse caso, usaremos a câmera IP Edimax 3116W, controlável por aplicativos móveis e web.

A Figura 3-12 mostra a aparência do dispositivo. Observe o ID da FCC na etiqueta na parte de trás.

Se consultarmos o ID FCC do dispositivo, que é NDD9530401309, no site [fccid.io], veremos a tela mostrada na Figura 3-13.

No site, podemos ver várias informações sobre o dispositivo, como alcance de frequência, acesso à configuração do laboratório, fotos internas, fotos externas, manual do usuário, procuração eletrônica (PoA) e muito mais.

Uma das coisas mais interessantes para se observar ao analisar as informações do FCC ID são as fotos internas do dispositivo. Você pode encontrá-los em <https://fccid.io/>.

A Figura 3-14 mostra as fotos internas do dispositivo. Um fato interessante a se notar é que, neste caso, as fotos também revelam que esta câmera IP possui uma interface UART, conforme sugerem os quatro “pads” mostrados na foto. Isso também é algo que exploraremos em nossos próximos capítulos para obter um shell root no dispositivo.

Como podemos ver, os IDs da FCC podem ser uma mina de ouro de informações e podem nos ajudar a aprender muitos detalhes sobre o dispositivo e seu funcionamento.

Outro fato interessante é que, às vezes, o fabricante deixa de solicitar um pedido de confidencialidade sobre informações confidenciais do dispositivo, como o esquema elétrico. O acesso ao esquema do dispositivo é extremamente útil, pois nos diz quais componentes eletrônicos diferentes são usados para construí-lo e nos ajuda a entender o dispositivo com muito mais profundidade.

**Pacote de Componente**

Um dos pontos que vale a pena mencionar, sempre que discutimos análise embarcada ou de hardware, é o tipo de encapsulamento. Sempre que você olhar para o interior de um dispositivo, verá vários componentes diferentes. Cada um dos componentes varia em tamanho, formato e outros aspectos com base na característica e funcionalidade do dispositivo.

Durante a fabricação e desenvolvimento de um dispositivo embarcado, existem várias opções de encapsulamento à sua escolha. Com base no encapsulamento que um componente está usando, para análise, precisaríamos de adaptadores de hardware correspondentes e outros componentes para interagir com eles. Os tipos de encapsulamento usados ​​com mais frequência estão listados aqui e mostrados na Figura 3-15.

1. DIL

Pacote único em linha

Pacote duplo em linha

TO-220

1. SMD

Cerpack

BGA

SOT-23

QFP

SOIC

SOP

**Chipsets de Rádio**

Uma das coisas adicionais importantes que você pode procurar nos dispositivos são os vários chipsets de rádio presentes. Esses chipsets podem lhe dar uma ideia de quais tipos de metodologias de comunicação um determinado dispositivo usa, mesmo que não esteja documentado ou mencionado em nenhum lugar.

Por exemplo, a Figura 3-16 é uma imagem interna de um Wink Hub que usa vários protocolos de comunicação, incluindo Wi-Fi, ZigBee e ZWave, além das interfaces de comunicação de hardware, como JTAG. [Imagem do Wink Hub mostrando os chips de rádio] (Figura 3-16 Wink Hub - chips de rádio)

**Conclusão**

Exploramos componentes de hardware adicionais em detalhes à medida que avançamos neste livro. No entanto, para um conhecimento profundo de vários componentes de hardware, eu recomendo fortemente que você dê uma olhada no livro Hardware Hacking de Nicholas Collins, disponível em [http://www.nicolascollins.com/texts/originalhackin gmanual.pdf](http://www.nicolascollins.com/texts/originalhackin%20gmanual.pdf)

**4.Comunicação UART**

Receptor Transmissor Assincrono Universal(UART-Universal Asynchronous Receiver/Transmitter) é um método de comunicação serial que permite que dois componentes diferentes em um dispositivo conversem entre si sem a necessidade de um clock. Iremos nos aprofundar em UART neste capítulo, pois é uma das interfaces de comunicação mais populares e possui grande importância na segurança IoT e em testes de penetração.

Também existe algo conhecido como Universal Synchronous/Asynchronous Receiver/Transmitter (USART), que transmite dados de forma síncrona e assíncrona dependendo da necessidade; no entanto, não vemos muitos dispositivos utilizando-o. Por esse motivo, não abordaremos USART e focaremos em UART.

Começaremos estabelecendo a base da comunicação serial e, em seguida, entraremos em detalhes sobre como identificar interfaces UART e interagir com elas. Este capítulo também serve como um capítulo introdutório para você iniciar sua jornada de exploração de hardware, caso ainda não tenha feito isso antes.

Ao concluir este capítulo, você será capaz de abrir um dispositivo, analisar os pinos UART possíveis, identificar as pinagens corretas e, por fim, se comunicar com o dispositivo alvo através de UART. Do ponto de vista da segurança, a capacidade de interagir com UART será útil para ler os logs de depuração de um dispositivo, obter acesso root não autenticado, acesso ao bootloader e muito mais.

**Comunicação em série**

Para qualquer dispositivo IoT ou embarcado, os diferentes componentes de um dispositivo precisam interagir entre si e trocar dados. A comunicação serial e a comunicação paralela são as duas maneiras pelas quais os componentes em um dispositivo trocam dados.

Como o próprio nome sugere, a comunicação serial é usada para transferir um bit por vez através de um determinado meio (veja a Figura 4-1), enquanto na comunicação paralela, um bloco de dados é transferido ao mesmo tempo, com cada bit exigindo um canal separado (e adicionalmente uma linha de referência - normalmente aterrada).

Como a comunicação paralela transfere um grande bloco de dados por vez, esse método requer várias linhas separadas para facilitar a comunicação. Como você pode imaginar, isso resultaria na necessidade de mais espaço na placa, o que geralmente não é desejável. Um protocolo de comunicação paralela é mostrado na Figura 4-2.

Essa é a razão pela qual a comunicação em série é um método de comunicação mais comum sempre que lidamos com dispositivos embarcados: ao contrário da comunicação paralela, ela requer apenas uma única linha para facilitar a troca de dados.

Alguns dos canais populares de comunicação em série dos quais você já deve ter ouvido falar são o Recommended Standard 232 (RS232), o Barramento Serial Universal (USB), o PCI, a Interface Multimídia de Alta Definição (HDMI), a Ethernet, a Interface Periférica Serial (SPI), o Circuito Integrado Inter (I2C), a Rede de Área do Controlador (CAN) e assim por diante.

O primeiro canal de comunicação em série usado foi o RS232, que oferecia uma taxa de transmissão de dados de 20 kbps; depois veio o USB 1.0, oferecendo taxas de 12 Mbit/s; seguido pelo USB 2.0, com uma velocidade de 480 Mbps; e finalmente o USB 3.0, com uma velocidade de 5 Gbps - quase 10 vezes mais rápido que seu predecessor. Também é importante notar que, devido aos recentes avanços tecnológicos, a comunicação serial está se tornando mais barata, mais rápida e mais confiável.

Agora que temos uma noção básica de comunicação serial e alguns de seus exemplos, vamos passar para o UART, que é o que abordaremos neste capítulo.

**O que, Por que e Como do UART**

Como descrito anteriormente, UART é um protocolo de comunicação serial assíncrona usado em muitos dispositivos embarcados e de IoT. Assíncrono significa simplesmente que, ao contrário de um protocolo síncrono (por exemplo, SPI), ele não possui um clock que sincroniza os dois dispositivos entre os quais a comunicação ocorre.

Os dados, no caso de UART, seriam transferidos sem a necessidade de uma linha adicional de clock externo (CLK). É por isso também que muitas outras precauções são tomadas ao transferir dados de forma assíncrona entre dispositivos em série para minimizar a perda de pacotes. Discutiremos a taxa de transmissão, o que tornará isso mais claro para você, nas seções posteriores deste capítulo.

**Pacote de Dados UART**

Um pacote de dados UART consiste em vários componentes:

1. **Bit de start:** O bit de start sinaliza que os dados UART virão a seguir. Normalmente é um pulso baixo (0) que você pode visualizar no analisador lógico.
2. **Mensagem:** A mensagem real que deve ser transferida em um formato de 8 bits. Por exemplo, se eu tiver que transmitir o valor A (com o valor 0x41 em hexadecimal), ele seria transferido como 0, 1, 0, 0, 0, 0, 0 e 1 na mensagem.
3. **Bit de paridade:** O bit de paridade não é tão relevante em cenários da vida real (com base na minha experiência), pois não vejo muitos dispositivos usando-o. Um bit de paridade é usado para realizar verificação de erro e corrupção de dados contando o número de valores altos ou baixos na mensagem e, com base em se é uma paridade ímpar ou par, indicaria que os dados não estão corretos. Lembre-se que o bit de paridade é usado apenas para verificação e validação de corrupção de dados, e não para correção real.
4. **Bit de stop:** O último bit que sinaliza que a mensagem concluiu a transmissão Normalmente, isso é feito por um pulso alto (1), mas também poderia ser feito por mais de um pulso alto, dependendo da configuração usada pelo desenvolvedor do dispositivo. Você pode entender muito melhor com a representação visual fornecida na Figura 4-3.

A maioria dos dispositivos que encontrei usa a configuração 8N1, o que significa oito bits de dados, nenhum bit de paridade e um bit de parada (veja a Figura 4-4). Podemos entender isso melhor se conectarmos um analisador lógico às interfaces UART do dispositivo. Um analisador lógico é um dispositivo que ajuda a exibir vários sinais e níveis lógicos de um circuito digital. É extremamente fácil de usar e simples de configurar com o protocolo ou comunicação que você está tentando analisar. Eu recomendaria obter um bom analisador lógico, como o Saleae Logic Analyzer ou Open Workbench Logic Sniffer, para todos os seus propósitos de análise lógica.

**Tipos de Portas UART**

Uma porta UART pode ser baseada em hardware ou software. Para lhe dar um exemplo, os microcontroladores AT89S52 e ATMEGA328 da Atmel possuem apenas uma porta serial de hardware. Se necessário, o usuário pode emular mais portas UART de software em pinos de entrada/saída de uso geral (GPIOs) específicos. Por outro lado, microcontroladores como LPC1768 e ATMEGA2560 possuem várias portas UART de hardware, todas as quais podem ser usadas para realizar análise e exploração baseadas em UART.

Mesmo que estejamos olhando para dispositivos do ponto de vista da segurança, uma das coisas a entender são as tecnologias que estamos discutindo - UART, JTAG, SPI, I2C e assim por diante; embora possam ser usadas para fins de pesquisa e exploração de segurança, sua função principal é para ... (insira a função principal das tecnologias aqui)

Seja para facilitar a comunicação entre componentes ou para fornecer funcionalidade adicional ao desenvolvedor.

UARTs baseadas em software são necessárias quando há a necessidade de conectar vários dispositivos via UART a um determinado dispositivo que possui apenas um conjunto limitado de pinos UART de hardware. Isso também oferece ao usuário a flexibilidade de usar os pinos GPIO como UART quando necessário e utilizá-los para outra finalidade em um momento posterior.

Não abordaremos UARTs baseadas em software em detalhes porque, em dispositivos comerciais do mundo real, normalmente não exigimos várias portas UART e não teremos a capacidade (ou acesso) para programar os GPIOs para emular UART, ou simplesmente porque não há pinos GPIO suficientes no nosso dispositivo alvo que possam ser emulados.

**Taxa de Baud**

Trabalhar com UART e discutir a não necessidade de CLK nos leva ao conceito de taxa de baud, que especifica a velocidade na qual os dados são transferidos entre dispositivos, ou, mais apropriadamente, o número de bits por segundo que estão sendo transferidos. Isso é necessário porque não há uma linha de clock no caso da comunicação UART, então ambas as comunicações precisam ter um entendimento mútuo da velocidade da comunicação de dados. É por isso que ambos os componentes concordarão em uma única taxa de baud durante todo o processo de troca de dados UART.

Durante qualquer exploração UART que você realize em sua jornada de pesquisa de segurança, uma das etapas iniciais sempre será identificar a taxa de transmissão do dispositivo alvo. Isso pode ser feito de várias maneiras, como observar a saída ao interagir pela serial em uma determinada taxa de transmissão e, se os dados parecerem ilegíveis, passar para a próxima taxa de transmissão. Para facilitar as coisas, existem alguns valores padrão de taxa de transmissão aos quais você encontrará a maioria dos dispositivos aderindo. As taxas de transmissão comuns são 9600, 38400, 19200, 57600 e 115200. Dito isso, o desenvolvedor de um dispositivo é livre para escolher um valor personalizado para a taxa de transmissão.

Para identificar a taxa de transmissão correta usando a abordagem mencionada, usamos um script escrito por Craig Heffner, chamado baudrate.py, disponível em <https://github.com/devttys0/baudrate/blob/master/baudrate.py>. Este script permite alterar as taxas de transmissão enquanto mantém uma conexão serial, para identificar facilmente qual é o valor correto da taxa de transmissão observando a saída e inspecionando visualmente qual taxa de transmissão fornece uma saída legível.

Antes de conectar a um dispositivo pela serial e identificar os valores da taxa de transmissão, vamos primeiro examinar as conexões de hardware que precisam ser feitas para interagir com o dispositivo alvo através da UART e explorá-lo.

**Conexões para Exploração UART**

Para realizar uma exploração baseada em UART, são necessários dois componentes principais:

* Dispositivo alvo: O dispositivo que deseja explorar.
* Dispositivo emulador de conexão serial: Um dispositivo que pode emular uma conexão serial para se comunicar com o dispositivo final, permitindo que o dispositivo alvo interaja com o seu sistema.

Hardware:

O hardware que utilizaremos neste exercício é o seguinte:

* Câmera IP Edimax 3116W: (Sinta-se à vontade para escolher qualquer outro dispositivo vulnerável que possua uma interface UART).
* Attify Badge: (Você também pode usar um USB-TTL normal ou BusPirate).
* Multímetro.
* Jumpers: (No caso de você querer soldá-los aos pads vazios para conectar os jumpers com firmeza).
* Fios jumper: Três unidades.

Para realizar as conexões, primeiro precisamos identificar onde fica a porta UART no dispositivo, ou quais são os pinos UART. Isso pode ser feito através de uma inspeção visual dos componentes internos do dispositivo, procurando por três ou quatro pinos ou pads próximos uns dos outros. Essa é uma maneira fácil de encontrar pinos UART, mas em alguns casos, você também pode encontrar dispositivos que possuem os pinos UART espalhados pela placa e não juntos em um único local. As Figuras 4-5 até 4-7 servem como referência para ajudá-lo a identificar portas UART em seu dispositivo alvo.

Agora que você identificou a localização dos pinos, vamos para a próxima etapa: identificar a função de cada pino individual. A UART é composta por quatro pinos que precisamos encontrar:

1. Transmitir (Tx): Envia dados do dispositivo para o outro dispositivo conectado.
2. Receber (Rx): Recebe dados do outro dispositivo para o dispositivo.
3. Terra (GND): Pino de referência de terra.
4. Tensão (Vcc): Tensão de alimentação, geralmente 3,3V ou 5V.

Para encontrar todos esses pinos, usamos um multímetro. Ele nos ajuda a identificá-los com base em dois testes:

* Continuidade (para GND): Verifica se há uma conexão elétrica com resistência muito baixa entre os pinos.
* Diferença de tensão (para os três pinos restantes): Mede a voltagem em cada pino para compará-la com a voltagem de referência (geralmente Vcc).

Um multímetro é uma combinação de voltímetro e amperímetro. Isso significa que ele permite observar o valor da tensão e da corrente durante a análise. Essa versatilidade é extremamente útil. Embora na maioria das vezes analisemos apenas a tensão, em alguns casos, pode ser necessário verificar a corrente que flui entre dois pinos diferentes para uma análise mais aprofundada. Vamos começar!

## **Identificando Pinos UART**

Como mencionado anteriormente, um multímetro é um dispositivo que pode medir tensão (V), corrente (A) e resistência (Ω). Por isso o nome multímetro - uma combinação de voltímetro e amperímetro.

Inicialmente, manteremos o dispositivo alvo desligado, pois realizaremos um teste de continuidade para identificar o pino terra (GND).

Para usar o multímetro, conecte os terminais como mostrado na Figura 4-8.

Agora que o multímetro está conectado, vamos prosseguir para encontrar os pinos UART de acordo com os passos descritos abaixo.

1. **Localizando o Pino Terra (GND):**

* Posicione a ponta de prova preta em uma superfície aterrada. Essa superfície pode ser qualquer parte metálica (por exemplo, a blindagem Ethernet do dispositivo) ou o pino GND do Attify Badge.
* Toque a ponta de prova vermelha em cada um dos pads individualmente. Repita esse processo com todos os pads até ouvir um bipe. O local onde você ouvir o bipe é o pino terra do dispositivo alvo. Certifique-se de que o dispositivo esteja desligado.
* É importante notar que pode haver vários pinos ou pads terra no dispositivo alvo, mas estamos interessados apenas no GND do par de pinos UART.
* Verifique se o seu multímetro está configurado como mostrado na Figura 4-9.

1. Localizando o pino Vcc:

* Coloque o multímetro novamente na posição V-20, pois agora vamos medir a tensão. Mantenha a ponta preta no pino GND e mova a ponta vermelha pelos outros pinos da UART (exceto o GND).
* Ligue e desligue o dispositivo. O local onde você vê uma tensão alta constante é o pino Vcc. Se não encontrar na primeira tentativa, ligue e desligue novamente.

1. Localizando o pino Tx:

* Reinicie o dispositivo novamente e meça a tensão entre os pinos restantes e o GND. Devido à grande quantidade de transferência de dados inicial durante a inicialização, você notará uma grande flutuação no valor da tensão durante os primeiros 10 a 15 segundos. Este pino será o pino Tx.

O Rx pode ser identificado pelo pino com a menor voltagem durante todo o processo, com a ponta preta do multímetro conectada ao GND do Attify Badge. Alternativamente, geralmente restará apenas um pino após esta etapa, que será o Rx.

Agora você deve ter conseguido identificar com sucesso todos os pinos diferentes presentes na UART do seu dispositivo alvo. Anote esses valores porque iremos usá-los ao fazer nossas conexões.

**Introdução ao Attify Badge**

Uma das ferramentas absolutamente necessárias no arsenal de um teste de penetração de IoT é um dispositivo capaz de trabalhar com diferentes protocolos de comunicação de hardware. A ferramenta que usaremos para todas as nossas necessidades de exploração de hardware é o Attify Badge.

O Attify Badge é uma ferramenta multifuncional que ajuda você a se comunicar com outros dispositivos IoT/embarcados por meio de várias interfaces de comunicação, como UART, SPI, I2C e até mesmo padrões como JTAG. Ele usa um chip FTDI que permite converter o protocolo de comunicação de hardware em uma linguagem compreendida por nossos sistemas. A Figura 4-10 mostra um Attify Badge.

A ferramenta possui um total de 18 pinos, dos quais 10 são para alimentação (3,3V e 5V) e aterramento. Estes pinos estão localizados na parte superior (primeiros nove pinos) e no canto inferior direito. Conforme ilustrado na Figura 4-11, os pinos D0 a D3 possuem uma função especial quando se trata de interagir com o hardware de dispositivos embarcados.

A tabela 4-1 apresenta a pinagem do Attify Badge para interação com diferentes protocolos de comunicação de hardware.

Tabela

Descrição gerada automaticamente

Para conectar o Attify Badge ao seu sistema, você precisará de um cabo micro USB. Se estiver usando o AttifyOS ou executando em um Mac OS, não serão necessárias ferramentas especiais para trabalhar com o dispositivo. No entanto, no Windows, pode ser necessário baixar o driver FTDI disponível em <https://ftdichip.com/drivers/> para que o dispositivo funcione com o seu sistema.

Para verificar se a conexão foi realizada com sucesso, você pode executar o comando lsusb (em uma máquina Linux). Esse comando deve listar um dispositivo como "Future Devices Technology International", que é o Attify Badge.

**Criando conexões finais**

Após identificar o esquema de pinos do dispositivo, o próximo passo é conectar os pinos UART do dispositivo ao UART do Attify Badge.

Os pinos do Attify Badge que nos interessam neste momento são D0 e D1, que representam transmissão (Tx) e recepção (Rx), respectivamente. O Tx (transmitir) da câmera IP deve ser conectado ao Rx (D1) do Attify Badge, e vice-versa para o Rx da câmera IP e Tx (D0) do Attify Badge. Utilize jumpers para realizar essas conexões.

Além disso, conecte o GND (terra) da câmera IP ao GND do Attify Badge. A tabela 4-2 resume as conexões para sua referência.



**Tabela

Descrição gerada automaticamente**

Lembre-se de não conectar o Vcc da câmera IP, pois isso poderia causar danos permanentes ao seu dispositivo. Vá em frente e faça as conexões entre as portas UART da câmera IP e o Attify Badge usando os jumpers, e conecte o Attify Badge ao seu sistema. A Figura 4-12 mostra a conexão final.

**Identificação da Taxa de Baud**

Como discutido anteriormente, a taxa de baud é a primeira coisa que devemos identificar sempre que realizamos uma exploração baseada em UART. Usaremos o script baudrate.py, conforme mencionado anteriormente. Antes de prosseguir, precisamos de uma informação adicional, que é a entrada do dispositivo Attify Badge conectado ao nosso laptop. Isso pode ser encontrado observando as entradas de /dev/, conforme mostrado na Figura 4-13.

Como você pode ver, nossa entrada está listada na imagem na porta COM /dev/ttyUSB0, que também é a porta COM padrão usada pelo baudrate.py. Vá em frente e execute-o:

git clone [https://github.com/devttys0/baudrate.git sudo python baudrate.py](https://github.com/devttys0/baudrate.git%20sudo%20python%20baudrate.py)

No início, você pode ver dados sem sentido assim que a câmera IP inicializar. Isso ocorre porque o dispositivo pode não estar configurado para transferir dados na taxa de transmissão padrão que o programa baudrate.py escolhe. Use as setas para cima e para baixo para navegar entre diferentes valores de taxa de transmissão. A taxa de transmissão em que você consegue ver caracteres legíveis é a taxa de transmissão correta do dispositivo. No nosso caso, a taxa de transmissão correta é 38400. Se você não vir nenhum dado, reinicie o dispositivo e verifique se as conexões estão corretas, com os valores corretos de Tx e Rx. É assim que se identifica a taxa de transmissão de qualquer dispositivo alvo.

**Interagindo com o Dispositivo**

Uma vez identificada a taxa de baud correta, o próximo passo é interagir com o dispositivo através da UART. Isso pode ser feito através do próprio script baudrate.py, pressionando Ctrl+C assim que ele detectar a taxa de baud correta, iniciando um programa chamado minicom. A outra maneira é iniciar manualmente um programa como screen ou minicom com as configurações identificadas.

Portanto, temos agora os dois valores de dados necessários:

1. Taxa de baud do dispositivo: 38400
2. Porta COM usada pelo Attify Badge: /dev/ttyUSB0

Vamos em frente e iniciar o programa screen com os valores fornecidos anteriormente.

sudo screen /dev/ttyUSB0 38400

Execute este comando e reinicie o dispositivo para poder ver os logs de debug da inicialização do dispositivo, conforme mostrado na Figura 4-14.

Se esperarmos mais alguns segundos para o dispositivo inicializar completamente e carregar o busybox, teremos um shell root não autenticado completo na câmera IP, conforme mostrado na Figura 4-15.

Parabéns! Agora você tem um shell root completo não autenticado no dispositivo. Você pode realizar várias ações aqui, como analisar o sistema de arquivos do dispositivo, modificar algumas configurações, identificar valores confidenciais ocultos, despejar o firmware e assim por diante.

Você também pode usar a ferramenta Attify Badge (Figura 4-16) disponível em <https://github.com/attify/attify-badge> para realizar esse processo usando uma interface gráfica do usuário (GUI).

Durante sua jornada de pesquisa de segurança de IoT, você ficará surpreso com o número de dispositivos comerciais do mundo real que concedem acesso root não autenticado ao dispositivo.

Alguns dos pontos a serem observados ao realizar a exploração UART são os seguintes:

* Certifique-se de que as conexões estão corretas; ou seja, Tx de um dispositivo vai para Rx do outro e Rx do outro vai para Tx.
* GND está conectado ao GND do outro dispositivo.
* Vcc não está conectado a nada.
* O valor da taxa de transmissão está identificado corretamente, caso contrário, você poderá ver dados ilegíveis.
* Certifique-se de usar um conversor de voltagem adequado ao usar um dispositivo serial de 3,3 V para um dispositivo serial de 5 V ou outros níveis de voltagem.

**Conclusão**

Neste capítulo, vimos como podemos começar a realizar exploração de dispositivos embarcados para dispositivos IoT usando comunicação serial, e especificamente, UART.

UART será útil para você em vários lugares, e você frequentemente encontrará dispositivos sem proteção, dando-lhe acesso a um shell root não autenticado pela UART.

Eu recomendo fortemente que você tente atividades adicionais depois de ter acesso UART, como interagir com o bootloader, modificar certos valores nas configurações, descobrir maneiras de despejar firmware pela UART e assim por diante.

**5. Exploração Usando I²C e SPI**

Neste capítulo, vamos dar uma olhada em dois dos outros protocolos seriais mais comuns (além de UART), nomeadamente I2C (pronunciado I-2-C ou I-quadrado-C) e SPI, e veremos como eles são úteis para nossa pesquisa de segurança e exploração de dispositivos IoT. SPI e I2C são protocolos de barramento úteis usados para comunicação de dados entre diferentes componentes em um circuito de dispositivo embarcado. SPI e I2C têm muitas semelhanças e algumas diferenças na forma como funcionam e como interagimos com eles.

Nós usamos principalmente técnicas de exploração SPI e I2C para despejar conteúdo (incluindo firmware e outros segredos confidenciais) do chip flash de um dispositivo, ou gravar conteúdo (por exemplo, uma imagem de firmware malicioso) no chip flash. Ambas as técnicas são extremamente úteis durante um teste de penetração ou ao realizar pesquisa de segurança em um dispositivo IoT. No entanto, como SPI e I2C são protocolos de barramento, você os encontrará em muitos outros lugares além de usá-los em Flash, como Relógios de Tempo Real (RTCs), LCDs, microcontroladores, conversores analógico-digital (ADCs) e assim por diante. Neste capítulo, porém, nos concentramos nos protocolos subjacentes e depois vemos como poderíamos usar esses protocolos para trabalhar com Flash e EEPROMs. Começamos com I2C e depois passamos para SPI, entendendo como podemos interagir com ambos e usá-los para nossos propósitos.

**I²C(Circuito Inter-Integrado)**

Vamos começar com um pouco de histórico para entender porque o I2C foi criado e como ele evoluiu. O I2C foi desenvolvido em 1982 pela Philips para permitir que seus chips se comuniquem e troquem dados com outros componentes. Na primeira versão do I2C, a velocidade máxima de transmissão de dados era de 100 kbps com um endereço de 7 bits, que mais tarde passou para 400 kbps com um endereço de 10 bits. Atualmente, componentes que utilizam I2C podem se comunicar entre si com uma velocidade de até 3,4 Mbps.

Analisando os aspectos técnicos do I2C, ele é um protocolo multimaster que requer apenas dois fios para permitir a troca de dados: dados seriais (SDA) e clock serial (SCL). No entanto, o I2C é half-duplex, o que significa que ele só pode enviar ou receber dados em um determinado momento.

**Por que não SPI ou UART?**

Pode parecer confuso no início: por que alguém usaria I2C em vez de UART ou SPI? Bem, há alguns motivos.

O desafio com UART é a limitação de facilitar a comunicação entre apenas dois dispositivos por vez. Além disso, como vimos no capítulo anterior, uma estrutura de pacote UART inclui um bit de início e parada, o que aumenta o tamanho geral do pacote de dados transferido, afetando também a velocidade de todo o processo. Além disso, a UART foi originalmente concebida para fornecer comunicação em longas distâncias, interagindo com dispositivos externos via cabos. Por outro lado, I2C e SPI são destinados à comunicação com outros periféricos localizados na mesma placa de circuito.

SPI é outro protocolo extremamente popular para transferência de dados entre componentes. SPI possui taxas de transmissão de dados mais rápidas em comparação com I2C, mas a única grande desvantagem do SPI é a necessidade de três pinos para transferência de dados e um pino para seleção de chip/escravo, o que aumenta a necessidade geral de espaço ao implementar o protocolo SPI para comunicação de dados em comparação com I2C.

**Serial Peripheral Interface**

SPI é um dos outros protocolos de comunicação mais populares usados em dispositivos embarcados. O SPI é full-duplex (diferente do I2C, que é half-duplex) e consiste em três fios - SCK, MOSI e MISO - e um chip select/slave select adicional. Nos casos em que não há dados para leitura, porém, quando há uma gravação acontecendo, o slave deve enviar dados dummy para que a conexão ocorra.

O SPI foi originalmente desenvolvido pela Motorola para fornecer comunicação serial síncrona full-duplex entre os dispositivos mestre e escravo. Ao contrário do I2C, no SPI apenas um único mestre controla todos os escravos e o mestre controla o clock para todos os escravos. A implementação geral e padronização do SPI é definida de forma bastante flexível e diferentes fabricantes podem modificar a implementação à sua maneira, devido à falta de um padrão rígido. Para entender a comunicação SPI para qualquer chip no dispositivo alvo, a melhor maneira é consultar a folha de dados e analisar como nosso alvo implementou o protocolo SPI para comunicação.

**Understanding EEPROM**

Tanto SPI quanto I2C são protocolos comuns quando se trata de falar sobre armazenamento de dados via Memória Apenas de Leitura Programável e Apagável Eletricamente (EEPROM). Nesta seção, veremos a EEPROM e entenderemos os vários pinos nela, o que será útil ao trabalhar com I2C e SPI. EEPROMs seriais normalmente possuem oito pinos, conforme listado na Tabela 5-1.

Tabela

Descrição gerada automaticamente

* Chip select (Seleção de chip): Como SPI e I2C (e outros protocolos) geralmente possuem vários dispositivos escravos, é necessário selecionar um escravo entre outros para qualquer ação. O pino chip select ajuda exatamente nisso - ajudando a selecionar uma EEPROM quando o #CS está baixo. Quando um dispositivo não é selecionado, não haverá comunicação entre o mestre e o escravo, e o pino de saída de dados seriais permanece em um estado de alta impedância.
* Clock (Relógio): O pino clock ou SCK (ou CLK) determina a velocidade com que a troca de dados e a comunicação devem ocorrer. O mestre é quem determina a velocidade do clock que os escravos devem obedecer. No entanto, no caso de I2C, os escravos podem modificar e diminuir a velocidade do clock se a velocidade do clock selecionada pelo mestre for muito rápida para os escravos. Esse processo também é conhecido como clock stretching (extensão de clock).
* MISO/MOSI: Como você deve ter imaginado, MISO e MOSI significam "master-in-slave-out" (mestre-entra-escravo-sai) e "master-out-slave-in" (mestre-sai-escravo-entra), respectivamente. Os pinos são usados de acordo com quem está enviando dados e quem está recebendo. No caso de I2C, como é half-duplex, ele só pode ler ou gravar dados em um determinado momento. No entanto, no caso de SPI, a leitura e a gravação de dados ocorrem simultaneamente. Se não houver dados para serem enviados (em leitura ou escrita), são enviados dados dummy (dados inúteis).
* Proteção contra escrita: Como o próprio nome sugere, este pino permite operações normais de leitura/escrita quando está alto. Quando o pino #WP está ativo baixo, todas as operações de gravação são inibidas.
* HOLD: Quando um dispositivo é selecionado e uma sequência serial está em andamento, o #HOLD pode ser usado para pausar a comunicação serial com o dispositivo mestre sem reiniciar a sequência serial. Para retomar a sequência serial, o pino #HOLD deve ser colocado em alto enquanto o pino SCK estiver em baixo.

Como já discutimos, o I2C funciona em duas linhas, chamadas SDA e SCL. A linha SDA é para troca de dados, enquanto a linha de clock, SCL, é controlada pelo mestre e determina a velocidade na qual a troca de dados ocorre. O mestre também armazena o endereço e o local da memória de todos os dispositivos escravos usados durante qualquer comunicação.

No I2C, ao contrário do SPI, pode haver vários mestres interagindo com vários escravos. Essa configuração é chamada de modo multimaster. Você pode se perguntar o que aconteceria se dois mestres quisessem assumir o controle de um barramento I2C ao mesmo tempo. A resposta é que qualquer mestre que puxar o SDA para BAIXO (0) ganhará o controle do barramento; ou seja, o valor zero vence a disputa.

**Explorando a 2C Security**

Agora que temos um bom entendimento dos conceitos fundamentais do I2C e como a transferência de dados acontece, vamos pular para como podemos explorar os dispositivos usando o protocolo I2C. Explorar o I2C aqui significa ler ou gravar dados do dispositivo usando uma EEPROM I2C em um dispositivo do mundo real.

Para esta seção, você pode escolher qualquer dispositivo que tenha um chip flash funcionando no protocolo de comunicação I2C. Para fins de demonstração, vou pegar o exemplo de um glicométer inteligente sem nome, que possui o recurso de salvar offline os registros de saúde do usuário no dispositivo. Para fins de exercícios práticos de laboratório, você pode obter qualquer dispositivo com EEPROM trabalhando em protocolo de comunicação I2C, como uma placa de breakout GY-521 ou mesmo qualquer chip I2C de <https://www.digikey.com/> e usar um adaptador EEPROM para se conectar a ele. As conexões permanecerão as mesmas mencionadas nas próximas seções, não importa qual dispositivo EEPROM I2C você escolher.

No caso do medidor inteligente de glicose que temos, ele usa um chip EEPROM MicroChip 24LC256, que funciona através do protocolo de comunicação I2C. A Figura 5-1 mostra uma ficha técnica online para o modelo I2C específico.

O primeiro passo para qualquer análise é encontrar o nome do componente na folha de dados e pesquisá-lo online. Com base na folha de dados do I2C encontrado neste dispositivo, que é uma EEPROM I2C Microchip de 256K organizada como memória serial de 32K × 8, descobrimos os pinos da EEPROM, conforme mostrado na Figura 5-2.

Certo, vamos analisar o que cada pino individual significa na Tabela 5-2.

Tabela

Descrição gerada automaticamente

**Estabelecendo conexões para exploração I2C com o Attify Badge**

Assim que tivermos as informações mencionadas anteriormente da folha de dados, podemos agora conectar a EEPROM ao nosso Attify Badge. Você pode conectá-la diretamente ao Attify Badge usando um clipe SOIC para segurar a EEPROM, ou removendo a EEPROM do dispositivo e soldando-a em um adaptador EEPROM correspondente à encapsulação da EEPROM.

A Figura 5-3 mostra como ficará a conexão entre a EEPROM e o Attify Badge, com o Attify Badge conectado ao nosso sistema.

* Para explicar melhor as conexões, veja o que está acontecendo na Figura 5-3.
* A0, A1, A2 e GND estão conectados ao GND.
* Vcc e WP estão conectados a 5V, pois a proteção contra gravação é ativa em nível baixo.
* D1 e D2 do Attify Badge estão conectados, o que corresponde à linha SDA.
* D0 está conectado à linha I2C SCL (clock).
* Depois de fazer todas as conexões, vamos dar uma olhada no script que usaremos para ler e gravar dados da EEPROM I2C.

**Entendendo o Código**

Para trabalhar com I2C, usaremos o script i2ceeprom.py de Craig Heffner localizado em <https://github.com/SjB/libmpsse>. Antes de realmente executar o script, vamos tentar entender como ele funciona. Isso também será útil se você quiser modificar o script para atender às suas próprias necessidades. Você também precisará modificar um pouco o script ao trabalhar com diferentes EEPROMs I2C com diferentes configurações e velocidades. O script começa mencionando o tamanho do chip EEPROM, que neste caso é de 32 KB, seguido pela especificação dos comandos de leitura e gravação. Ele também especifica posteriormente a velocidade como 400 KHz, conforme mostrado em nossa folha de dados. Observe que diferentes EEPROMs I2C podem ter velocidades diferentes e você precisará modificar este valor para se adequar ao seu alvo.

Texto

Descrição gerada automaticamente

Em seguida, tentamos iniciar o clock I2C usando eeprom.Start() e enviando o comando Start para inicializar a EEPROM na velocidade de 400 kHz.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Agora, se quisermos ler dados da EEPROM, o script primeiro verifica se a EEPROM está disponível procurando por um ACK de start(). Em seguida, envia o comando Ler usando eeprom.Write(RCMD) e configura a EEPROM para o modo de leitura. Uma vez que tudo esteja configurado, ele simplesmente começa a ler o conteúdo da EEPROM e salva-o em data().

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Após a operação de leitura ser concluída, fechamos a conexão I2C e escrevemos o conteúdo em um arquivo chamado EEPROM.bin.

Texto

Descrição gerada automaticamente

Depois que executarmos o script após fazer as conexões apropriadas, veremos que o conteúdo da EEPROM foi despejado no arquivo (veja a Figura 5-4).

De forma similar, também podemos gravar dados no chip I2C.

É assim que realizamos a análise I2C em qualquer dispositivo. Para resumir, estas são as etapas envolvidas no processo:

* Abrir o dispositivo.
* Identificar o chip I2C na placa de circuito impresso (PCB).
* Anotar o número do componente impresso no chip I2C.
* Pesquisar online pela ficha técnica para determinar a pinagem.
* Fazer as conexões necessárias.
* Utilizar o script i2ceeprom.py para ler ou gravar dados na EEPROM I2C.

**Mergulhando no SPI**

Já que entendemos I2C e EEPROM, vamos nos aprofundar em como o SPI funciona e como interagir com dispositivos alvo usando o protocolo de comunicação SPI.

Um mestre SPI se comunica com seus escravos usando quatro linhas:

* Serial clock (SCK): Relógio serial
* Master-out-slave-in: Mestre para escravo (entrada)
* Master-in-slave–out: Mestre (entrada) para escravo (saída)
* Slave select (SS; active low, saída do mestre): Seleção de escravo (nível baixo ativo, saída do mestre)

Dentre esses sinais, os pinos SCK, MISO e MOSI são compartilhados pelos dispositivos escravos, enquanto cada escravo SPI terá sua própria linha SS exclusiva. No SPI (diferente do I2C), pode haver apenas um mestre e vários escravos. O mestre no SPI é o responsável pelo clock. Para lhe dar uma melhor perspectiva de como é uma conexão SPI, a Figura 5-4 fornece um diagrama de comunicação SPI com um único mestre e vários escravos. Discutiremos os pinos individualmente à medida que nos aprofundamos no SPI.

A velocidade do SPI não é limitada, e é por isso que interfaces SPI são geralmente mais rápidas do que outros protocolos. Além disso, o fato de ser full-duplex a torna a melhor escolha para desenvolvedores de dispositivos que desejam aproveitar a velocidade.

**Como o SPI funciona**

Primeiramente, o mestre configura a frequência do clock de acordo com a frequência do clock do dispositivo escravo, normalmente até alguns MHz. A velocidade máxima do clock que podemos ter em SPI é metade da velocidade do clock do mestre. Por exemplo, se o mestre opera a 32 MHz, a taxa máxima do clock serial pode ser de até 16 MHz.

Para iniciar a comunicação, o mestre seleciona o dispositivo escravo com um nível lógico 0 na linha SS. Lembre-se que a cada ciclo de clock ocorre uma transmissão de dados full-duplex.

O mestre inicia a comunicação enviando um bit na linha MOSI, que é lido pelo escravo, enquanto o escravo envia um bit na linha MISO, que é lido pelo mestre. O bit mais significativo (MSB) é deslocado primeiro e um novo bit menos significativo (LSB) é deslocado para o mesmo registrador. Uma vez que o bit do registrador tenha sido deslocado para fora e para dentro, o mestre e o escravo trocaram o valor do registrador com sucesso.

**Leitura e escrita em EEPROM via SPI**

I wasn't able to translate the content of the webpage using the browsing extension, but here is a general translation to portuguese of the provided text:

Para ler e gravar dados de ou para uma EEPROM SPI, usamos uma ferramenta chamada spiflash.py disponível para download em <https://github.com/devttys0/libmpsse/>. Depois de clonar o repositório, basta navegar até a pasta src/examples/, onde você encontrará o script spiflash.py, que é o que vamos usar. Spiflash.py primeiro define vários valores padrão do protocolo SPI, como o comando de leitura e gravação usado pela maioria dos chips que usam SPI. Observe que SPI é um protocolo bastante flexível, o que significa que os desenvolvedores podem definir seus próprios valores personalizados de leitura e gravação. Nesse caso, precisamos modificar os valores mostrados no código a seguir.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente



Em seguida, definimos uma velocidade padrão na qual o script irá interagir com o chip alvo através do SPI. Neste caso, é 15 MHz; no entanto, podemos alterar a velocidade usando o parâmetro "-f" durante a execução do script. O script então continua e também define os pinos WP e HOLD como alto na seção \_init\_gpio.

Texto, Carta

Descrição gerada automaticamente

Em seguida, temos os blocos de código de Leitura, Gravação e Apagar. Neste trecho, o script se conecta ao chip alvo através de SPI usando a biblioteca mpsse e realiza operações de escrita, leitura e apagamento utilizando os flags fornecidos durante a execução (runtime) e definidos anteriormente no código (WCMD, RCMD, WECMD, RECMD).

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança baixa

Texto, Carta

Descrição gerada automaticamente

Na próxima seção de código, as várias flags que podemos usar com o código são mencionadas, conforme mostrado aqui.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Como você pode ver, podemos definir valores como especificar se queremos ler, escrever ou apagar, além de fornecer o tamanho para leitura e escrita, endereço inicial para realizar a operação e frequência de clock personalizada em vez do padrão de 15 MHz. Também temos a opção -v, que verifica se os dados que foram gravados ou lidos do chip são iguais aos originais.

Agora que estamos familiarizados com o script, podemos ir em frente e experimentá-lo em um dispositivo de destino. No meu caso, tenho um flash SPI Winbond que removi da PCB por dessoldagem. Uma vez dessoldado, podemos então soldá-lo em um adaptador EEPROM (ou leitor). Você também pode lê-lo diretamente enquanto o chip está no dispositivo, conectando miniprobes ao EEPROM ou usando um clipe SOIC de um dispositivo IoT real sem remover o chip do dispositivo.

A Figura 5-6 mostra como fica nosso flash SPI quando soldado ao adaptador EEPROM.

Vamos começar fazendo todas as conexões necessárias para SPI. Para fazer isso, precisamos primeiro entender a pinagem do nosso chip flash SPI alvo, que no nosso caso é o W25Q80DVSNIG. Se procurarmos online por folhas de dados para este chip flash, encontraremos a pinagem mencionada na folha de dados, conforme mostrado na Figura 5-7.

O próximo passo seria fazer as conexões necessárias usando o Attify Badge ou qualquer hardware compatível baseado em FTDI. Para descobrir onde a numeração dos pinos começa no chip real, em comparação com o datasheet, observe o entalhe na seção superior esquerda do chip e use-o para contar os números dos pinos. A Tabela 5-3 mostra a pinagem do Attify Badge para SPI.

Tabela

Descrição gerada automaticamente

Agora que sabemos as pinagens, estas são as conexões que precisamos fazer para se comunicar usando SPI:

* Conecte CLK em SCK (D0).
* Conecte MOSI/DO em MISO (D1). (Observe a inversão aqui, comparado com o inglês)
* Conecte MISO/DI em MOSI (D2). (Observe a inversão aqui, comparado com o inglês)
* Conecte CS em CS (D3).
* Conecte WP, HOLD e Vcc a 3,3V.
* Conecte GND a GND.

Uma das coisas que precisamos observar aqui é que as conexões de MOSI e MISO serão invertidas se você estiver usando outra ferramenta em vez de Attify Badge (por exemplo, Bus Pirate). Isso ocorre devido às convenções de nomenclatura do Attify Badge.

Depois de ter todas as conexões feitas, podemos agora prosseguir e executar o script spiflash.py e tentar ler dados do SPI EEPROM. A sintaxe do spiflash.py é mostrada aqui.

Tela de celular com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

Agora que sabemos as pinagens, estas são as conexões que precisamos fazer para se comunicar usando SPI:

Como podemos ver na Figura 5-8, conseguimos ler com sucesso o conteúdo do chip EEPROM flash SPI e armazená-lo em nosso sistema local.

Isso significa que agora, com qualquer dispositivo, você poderá despejar o conteúdo que ele vem armazenando em seu chip EEPROM. Além disso, também podemos gravar dados no chip, conforme mostrado na Figura 5-9.

Isso é extremamente útil, pois podemos gravar uma versão modificada do firmware de um dispositivo se formos capazes de interagir com o chip de memória flash EEPROM através de SPI.

**Descartando firmware usando SPI e Attify Badge**

Vamos tentar em outro dispositivo que tenha um firmware completo (nesse caso, OpenWRT) e extrair o firmware usando o script spiflash.py e o Attify Badge. O dispositivo que usaremos neste caso é um WRTNode, mostrado na Figura 5-10. Para fins de exercício prático, você pode usar o mesmo dispositivo (WRTNode) ou qualquer outra placa de desenvolvimento que tenha uma interface SPI.

Podemos observar que o WRTNode possui vários pinos e pads que nos permitem conectá-lo e interagir com ele. Como é uma placa de desenvolvimento popular, podemos encontrar o datasheet do WRTNode online (Figura 5-11).

Agora que já estamos familiarizados com o protocolo de comunicação SPI e como interagir com dispositivos usando esse protocolo com o Attify Badge, podemos interagir com o WRTNode. Nesse caso, se lermos os dados do chip flash, será todo o firmware, do qual poderemos extrair o sistema de arquivos. Embora abordemos a análise de firmware e a extração do sistema de arquivos no Capítulo 7, mostrarei aqui brevemente o processo de despejar o firmware de um dispositivo usando SPI.

A Figura 5-12 mostra uma visão tabular das conexões no caso do WRTNode, que é igual ao que vimos anteriormente.

Depois de ter feito as conexões, a Figura 5-13 mostra o diagrama de conexão final para maior clareza

Após a conexão, a próxima etapa seria a mesma de antes, que é executar spiflash.py e então especificar um tamanho grande o suficiente para que todo o conteúdo do chip flash seja despejado. A Figura 5-14 mostra o processo de dumping do firmware.

Finalmente, assim que tivermos o wrtnode-dump.bin, poderemos executar ferramentas de análise de firmware (que abordaremos mais tarde) como o Binwalk e obter todo o sistema de arquivos original (Figura 5-15).

É assim que aplicamos os conjuntos de habilidades de exploração SPI em dispositivos do mundo real para despejar todo o firmware do dispositivo.

**Conclusão**

Neste capítulo, abordamos vários tópicos, incluindo EEPROM, I2C e SPI. Também vimos como podemos ler e gravar dados da EEPROM usando os protocolos de comunicação I2C e SPI. Este conhecimento será extremamente útil para você quando estiver realizando pentesting em um dispositivo do mundo real e quiser olhar para o firmware, que seria armazenado na EEPROM, ou qualquer informação confidencial.

Você também pode modificar uma imagem de firmware despejada do chip EEPROM, gravá-la de volta e analisar o comportamento do dispositivo.

No próximo capítulo, começamos a olhar para um dos outros conceitos mais populares em hacking de dispositivos embarcados, JTAG.

Freqüentemente, você encontrará dispositivos comerciais do mundo real armazenando conteúdo como chaves secretas, firmware, binários e partes de dados interessantes em sua flash EEPROM, que com o conhecimento adquirido neste capítulo, podem ser explorados.

1. **Debugando e Exploração de JTAG**

Nos capítulos anteriores, vimos vários protocolos de comunicação, como UART, SPI e I2C. Neste capítulo, abordamos JTAG, que é um pouco diferente do que vimos até agora, e não é exatamente um protocolo de comunicação. JTAG é um conceito amplamente mal compreendido, mesmo dentro da comunidade de segurança.

O Joint Test Action Group (JTAG) é uma associação criada em meados da década de 1980, quando um grupo de empresas se uniu para resolver o problema de depuração e teste de chips ao lidar com a crescente complexidade dos dispositivos.

Durante esse período, os fabricantes de dispositivos embarcados perceberam o problema no teste tradicional de placa de pregos com os novos PCBs montados, devido ao aumento da densidade do dispositivo, especialmente ao trabalhar com chips com um número extremamente alto de pinos. Imagine o esforço manual necessário para testar centenas de chips com vários pinos em cada um deles e verificar se cada um deles está funcionando bem e se comunicando adequadamente. Para superar esse problema, os fabricantes criaram um padrão que lhes permite embutir uma peça de hardware no próprio chip para permitir um teste mais fácil de vários pinos presentes em diferentes chips do PCB. Essa metodologia foi padronizada pelo IEEE em 1990 e chamada de IEEE 1149.1.

JTAG não é um padrão ou protocolo, mas sim apenas uma forma de testar e depurar diferentes chips presentes no dispositivo. O JTAG usa uma técnica conhecida como varredura de limite, que permite aos fabricantes testar e diagnosticar PCBs montados com muito mais facilidade em comparação com a antiga abordagem tradicional.

**Teste de Limite**

Como mencionado, teste de limite é uma técnica para depurar e testar vários pinos de diferentes chips presentes em um circuito. Isso é feito pela adição de um componente chamado células de teste de limite próximo a cada pino do chip que precisa ser testado. Os vários pinos de E/S do dispositivo são conectados em série para formar uma cadeia. Esta cadeia pode então ser acessada pelo que é chamado de porta de acesso de teste (TAP).

O teste de limite ocorre enviando dados para um dos chips e comparando a saída com a entrada para verificar se tudo está funcionando corretamente. Cada chip na Figura 6-1 está conectado em série.

Diagrama, Desenho técnico

Descrição gerada automaticamente

Observe os pads de E/S e as células de análise de contorno próximas à periferia de cada chip. Essas células de análise de contorno podem ser acessadas e verificadas quanto aos valores nos pinos associados a elas. Um arquivo externo conhecido como arquivo de linguagem de descrição de análise de contorno define os recursos da lógica de análise de contorno de qualquer dispositivo individual.

**Instruções dos testes de limite**

Defino um conjunto de instruções pela norma IEEE 149.1 que devem ser disponibilizadas para um dispositivo em caso de uma varredura de limite. Essas instruções estão listadas aqui.

* BYPASS: A instrução BYPASS coloca o registrador BYPASS na cadeia DR, de forma que o caminho do TDI e TDO envolva apenas um único flip-flop (registrador de deslocamento). Isso permite que um chip específico seja testado em uma cadeia serial sem nenhuma sobrecarga ou interferência de outros chips.
* SAMPLE/PRELOAD: A instrução SAMPLE/PRELOAD coloca o registrador de varredura de limite na cadeia DR. Esta instrução é usada para pré-carregar os dados de teste no registrador de varredura de limite (BSR). Também é usado para copiar o valor de E/S do chip para o registrador de dados, que pode então ser movido para fora em estados DR de deslocamento sucessivos.
* EXTEST: A instrução EXTEST permite ao usuário testar o circuito externo. É como SAMPLE/PRELOAD, mas também direciona o valor do registrador de dados para os pads de saída.

**Teste de processo**

Para lhe dar uma visão geral clara, veja como seria o processo geral de teste para um processo de varredura de limite:

* O controlador TAP aplica dados de teste nos pinos TDI.
* O BSR monitora a entrada para o dispositivo e os dados são capturados pela célula de varredura de limite.
* Os dados então entram no dispositivo através dos pinos TDI.
* Os dados saem do dispositivo através dos pinos TDO.
* O testador pode verificar os dados no pino de saída do dispositivo e confirmar se tudo está funcionando.

Esses testes podem ser usados para encontrar desde um simples defeito de fabricação até componentes ausentes em uma placa, pinos desconectados ou posicionamento incorreto do dispositivo e até mesmo condições de falha do dispositivo.

**Debugando com JTAG**

Mesmo que o JTAG tenha sido originalmente criado para auxiliar na eliminação do antigo teste de cama de pinos, no mundo do desenvolvimento e teste embarcado da nova era, ele é usado para realizar diversas atividades, como depurar os vários chips presentes no dispositivo, acessando valores de pinos individuais em cada chip, teste geral do sistema, identificando componentes defeituosos em uma PCB altamente densa e assim por diante. Como o JTAG está disponível nos sistemas desde o início, assim que o sistema é inicializado, ele se torna extremamente útil para testadores e engenheiros analisarem todos os vários componentes presentes no dispositivo embarcado.

Para testadores de penetração e pesquisadores de segurança, é extremamente útil, pois nos permite depurar o sistema de destino e seus componentes individuais. Isso também significa que se nossa placa alvo tiver acesso JTAG disponível e contiver um chip flash onboard, poderíamos despejar o conteúdo do chip flash via JTAG. Também poderemos definir pontos de interrupção e analisar toda a pilha, conjuntos de instruções e registradores durante a depuração com JTAG, integrando-o a um depurador.

Agora que você sabe o quão útil JTAG será para identificarmos vulnerabilidades no dispositivo alvo ou realizar pesquisas de segurança, o próximo passo seria identificar os pinouts JTAG presentes no dispositivo alvo.

**Identificando Pinagens JTAG**

Identificar as pinagens JTAG pode ser um pouco mais complicado do que UART, onde tudo o que você precisa é procurar um conjunto de três ou quatro pinos e depois usar um multímetro para identificar as pinagens individuais. No caso de JTAG, precisaremos usar ferramentas adicionais (por exemplo, JTAGulator) para determinar efetivamente as pinagens individuais presentes em nosso dispositivo alvo.

Outra coisa a se observar ao trabalhar com JTAG é que na maioria dos dispositivos você encontrará as esferas JTAG, em vez de pinos JTAG ou esferas com furos, o que também torna importante ter um pouco de experiência em soldagem se quisermos explorar dispositivos do mundo real via JTAG.

Em JTAG, geralmente estamos preocupados com quatro pinos:

* TDI.
* TDO.
* TMS.
* TCK.

Antes de olharmos para as pinagens JTAG, vamos dar uma olhada em como as pinagens JTAG possivelmente se parecem, para que seja mais fácil localizá-las em uma placa de circuito dada. As Figuras 6-2 a 6-4 mostram alguns exemplos de pinagens JTAG. A Figura 6-2 mostra uma interface JTAG de 14 pinos no roteador Netgear WG602v3.

**Circuito eletrônico azul

Descrição gerada automaticamente com confiança média**

**Tela de jogo de vídeo game

Descrição gerada automaticamente com confiança média**

**Circuito eletrônico com fios

Descrição gerada automaticamente**

As figuras 6-3 e 6-4 mostram a imagem da placa de circuito impresso (PCB) do Wink Hub e as diferentes interfaces JTAG.

Agora que você entende como as pinagens JTAG podem aparecer em um dispositivo do mundo real, vamos em frente e começar a identificar qual é a pinagem da interface JTAG que acabamos de encontrar.

Para este exercício, você pode usar qualquer dispositivo com uma interface JTAG. Para começar, porém, eu recomendaria escolher uma placa de desenvolvimento que tenha uma interface JTAG especificada nela que possa ser usada para depuração. Bons exemplos são Raspberry Pi ou Intel Galileo, ambos com pinos JTAG.

Podemos identificar pinagens JTAG usando duas abordagens, que diferem de acordo com o hardware usado:

1.Usando JTAGulator.

2.Usando Arduino flashado com JTAGEnum.

**Usando JTAGulator**

JTAGulator é uma ferramenta de hardware open source, projetada por Joe Grand da Grand Idea Studios, que nos auxilia na identificação de pinos JTAG para um determinado dispositivo alvo. Possui 24 canais de I/O que podem ser utilizados para descoberta de pinagem e também para detecção de pinagem UART.

Ele usa um chip FT232RL que permite lidar com todo o protocolo USB em um único chip e nos possibilita apenas conectar o dispositivo e fazê-lo aparecer como uma porta serial virtual com a qual podemos interagir usando uma tela ou minicom. A Figura 6-5 mostra uma imagem do JTAGulator.

Uma imagem contendo relógio, medidor

Descrição gerada automaticamente

Para usar o JTAGulator, precisamos conectar todos os pinos do nosso dispositivo alvo aos canais do JTAGulator, enquanto conectamos o terra ao terra. Feito isso, basta conectar o JTAGulator ao nosso sistema e executar uma tela com uma taxa de transmissão de 115200.



Uma vez na tela do JTAGulator, o próximo passo é definir a tensão do sistema alvo pressionando V para selecionar a tensão alvo. Após selecionar a tensão, o próximo passo é selecionar uma varredura BYPASS para encontrar os pinos. Ao selecionar isso, você deverá especificar quantos canais selecionou para os pinos.

Depois de ter selecionado tudo, o JTAGulator detectará os vários pinos JTAG para você, conforme mostrado na Figura 6-6.

Texto, Carta

Descrição gerada automaticamente

Com base em quais pinos do seu dispositivo alvo estão conectados a qual canal, você poderá identificar a pinagem da interface JTAG no dispositivo alvo.

**Usando Arduino Flashnado com JTAGEnum**

Outra técnica popular para identificar a interface JTAG é usar o Arduino. Esta é uma alternativa muito mais barata em comparação com o JTAGulator, mas existem algumas limitações, como a varredura ser extremamente lenta e não ter a capacidade de detectar pinagens UART como o JTAGulator faz.

Para usar o JTAGEnum com Arduino, o primeiro passo é usar o programa JTAGEnum disponível em <https://github.com/cyphunk/JTAGenum>.

Depois de ter o código de exemplo, abra o ambiente de desenvolvimento integrado (IDE) do Arduino e cole o código na janela do editor, conforme mostrado na Figura 6-7. Selecione a porta correta e o tipo de Arduino nas opções do menu. No nosso caso, temos um Arduino Nano conectado ao nosso sistema. Clique no botão Carregar localizado no canto superior direito e você verá que o código foi carregado.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Agora que carregamos o código no nosso Arduino, o próximo passo é se comunicar com o Arduino através de uma conexão serial. Isso pode ser feito através do Monitor Serial presente no IDE Arduino ou usando um programa como o screen ou minicom, como mostrado na Figura 6-8.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Assim que tivermos o código JTAGEnum instalado e funcionando, podemos pressionar "s" para iniciar a varredura de várias combinações e identificar as pinagens JTAG. Esse processo pode levar um pouco mais de tempo do que o JTAGulator, mas no final informará as pinagens JTAG dos vários fios conectados ao Arduino, conforme mostrado na Figura 6-9.

Tabela

Descrição gerada automaticamente com confiança média

Assim como você fez anteriormente, mapeie esses fios para os conectados no dispositivo alvo e você terá as pinagens JTAG reais do dispositivo alvo.

Agora que identificamos as pinagens JTAG do dispositivo alvo, a próxima etapa é conectar-se à interface JTAG e depurar o dispositivo alvo e os programas nele executados. Para isso, precisamos de conhecimento do OpenOCD, que é o que discutiremos na próxima seção.

**openOCD**

openOCDé uma ferramenta que nos permite realizar depuração on-chip com nosso dispositivo alvo via JTAG.openOCD, desenvolvido por Dominic Rath, é um software open source que se comunica com a porta JTAG de um depurador de hardware. A seguir estão algumas das coisas que podemos fazer com a depuração JTAG:

* Depurar os vários chips presentes no dispositivo.
* Definir breakpoints e analisar registradores e pilha em um determinado momento.
* Analisar memórias flash localizadas no dispositivo.
* Programar e interagir com as memórias flash.
* Despejar firmware e outras informações confidenciais.

openOCD, como você pode ver, é uma ferramenta extremamente útil quando temos que trabalhar com JTAG. Nas próximas seções, veremos como podemos configurar o openOCD e usá-lo para realizar a exploração adicional do nosso dispositivo alvo.

**Instalando Software para JTAG**

Algumas das ferramentas que usaremos para depurar JTAG são:

* openOCD
* GDB-Multiarch
* Ferramenta Attify Badge

Instalar o openOCD no seu sistema é simples se feito usando o apt e pode ser feito executando este comando:

Tela de celular com aplicativo aberto

Descrição gerada automaticamente com confiança média

Uma imagem contendo Texto

Descrição gerada automaticamente

Após instalar o OpenOCD, estaremos prontos para iniciar nosso processo de exploração.

Uma utilidade adicional e interessante para instalar é o GDB-Multiarch. Ele permite usar o GDB para depurar binários de arquiteturas diferentes, pois na maioria das vezes lidaremos com dispositivos e binários alvo que não são feitos para a arquitetura x86 comum.

Como alternativa, se você instalar a ferramenta Attify Badge de <https://github.com/attify/attify-badge> e executar o install.sh, ele instalará automaticamente todas as ferramentas necessárias, incluindo o OpenOCD. Você também pode usar o AttifyOS localizado em <https://github.com/adi0x90/attifyos>, que já vem configurado com todas as ferramentas exigidas.

**Hardware para debugar JTAG**

Pelo lado do hardware, a depuração e exploração JTAG podem ser feitas com as seguintes ferramentas:

* Attify Badge ou outras ferramentas como Bus Pirate ou Segger J-Link.
* Dispositivo alvo com a interface JTAG.

Para simplificar, estamos usando um Attify Badge para fins de depuração JTAG. Para usar o Attify Badge com o dispositivo alvo, precisaremos conectar os pinos JTAG correspondentes do dispositivo com os pinos JTAG do Attify Badge, que veremos na próxima seção.

Para utilizar o Attify Badge (ou qualquer outro hardware), precisamos do arquivo de configuração OpenOCD para ele e do arquivo de configuração para o dispositivo alvo (e para qualquer outro dispositivo na cadeia). Aqui, o Attify Badge funcionará como um adaptador JTAG, e o dispositivo alvo pode ser um processador ou um controlador.

Antes de iniciar as conexões para JTAG, precisamos verificar se o controlador do nosso dispositivo alvo é compatível com o OpenOCD. Para fazer isso, podemos verificar a tabela da lista de dispositivos fornecida junto com o código-fonte do OpenOCD, conforme mostrado na Figura 6-10.

**Texto preto sobre fundo branco

Descrição gerada automaticamente**

Você sempre deve se certificar de que seu dispositivo alvo esteja listado nas opções do OpenOCD, que vêm junto com o código fonte. Caso contrário, será necessário criar um arquivo de configuração manual para o seu dispositivo.

**Configurando o ambiente para debugar JTAG**

Now that we have everything in place, we need to make the connections required for JTAG debugging. The Attify Badge pinouts for JTAG are provided in Table 6-1.

**Tabela

Descrição gerada automaticamente**

Assim que descobrirmos a conexão, o próximo passo é descobrir a pinagem para a nossa placa alvo e fazer as conexões. As conexões seriam as seguintes:

* TCK (D0) vai para CLK do dispositivo alvo.
* TDI (D1) vai para TDI do dispositivo alvo.
* TDO (D2) vai para TDO do dispositivo alvo.
* TMS (D3) vai para TMS do dispositivo alvo.

Os pinos que funcionam como CLK, TDI, TDO e TMS irão diferir com base no processador ou controlador do dispositivo alvo que você está tentando explorar.

Para fins de demonstração, pegaremos um dispositivo com a família de microcontroladores STM32F103C8. O diagrama de pinagem retirado da sua ficha técnica é exibido para melhor compreensão na Figura 6-11.

Diagrama

Descrição gerada automaticamente

Após concluir as conexões, o próximo passo é garantir que tenhamos os arquivos de configuração (.cfg) para o Attify Badge, bem como para o nosso destino.

O arquivo de configuração do Attify Badge, badge.cfg, está disponível nos downloads do código do livro, conforme mostrado aqui:

Uma imagem contendo Logotipo

Descrição gerada automaticamente

Uma imagem contendo Forma

Descrição gerada automaticamente

Para o arquivo de configuração do nosso alvo, o microcontrolador STM32, podemos obtê-lo a partir das próprias configurações do OpenOCD. A Figura 6-12 mostra uma representação gráfica das conexões atuais.

Diagrama

Descrição gerada automaticamente

Após as conexões serem feitas, podemos executar o seguinte comando para verificar se podemos usar o OpenOCD para depurar o alvo, como mostrado aqui.

Texto

Descrição gerada automaticamente

Texto, Carta

Descrição gerada automaticamente

Como podemos ver nesse texto, o OpenOCD é capaz de se conectar ao nosso dispositivo alvo e nos mostra informações adicionais, como seis pontos de interrupção, quatro pontos de vigilância e muito mais.

Depois de chegar a essa tela, você pode usar o telnet para se comunicar com a instância OpenOCD, que se conectou ao nosso dispositivo alvo por JTAG.

Texto, Carta

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

Como você pode ver, conseguimos conectar ao nosso dispositivo e chip alvo através de JTAG usando o OpenOCD. Isso significa que tivemos sucesso em identificar as pinagens JTAG corretas e agora podemos prosseguir com a exploração do nosso dispositivo alvo.

**Explorando JTAG**

Agora que você está conectado com sucesso, neste ponto você pode usar a sessão telnet com OpenOCD e JTAG para gravar firmware no microcontrolador, depurar binários e até mesmo despejar firmware dele.

Vamos dar uma olhada neles um por um, começando pela gravação do firmware no dispositivo.

**Escrita de Dados e Firmware em um Dispositivo**

Como mencionado anteriormente, o JTAG pode ser usado para gravar firmware no dispositivo.

Isso é útil quando você está avaliando o dispositivo alvo e deseja gravar uma versão modificada do firmware para contornar as restrições de segurança do dispositivo. Para gravar um novo firmware no dispositivo, vamos primeiro verificar o endereço onde a memória flash começa. Em seguida, podemos usar este endereço para gravar um novo firmware no dispositivo.

Texto

Descrição gerada automaticamente

A memória flash, neste caso, começa no endereço 0x08000000 e o tamanho atual do conteúdo nesse endereço é 0x0, o que indica que o dispositivo de destino não contém firmware no momento. usamos o endereço da saída anterior e o passamos para o nosso próximo comando, especificando a gravação de um firmware personalizado, firmware.bin. O firmware, neste caso, permite a autenticação via UART para a nossa placa alvo.

Para gravar o firmware no dispositivo de destino, use o seguinte comando:

Texto

Descrição gerada automaticamente

Como você pode ver, a gravação do firmware foi concluída com sucesso. Podemos verificar isso realizando um "flash banks" e observando a alteração no tamanho de armazenamento da memória flash.

Texto

Descrição gerada automaticamente

Esta técnica é útil ao trabalhar com dispositivos onde você deseja despejar o conteúdo de vários chips de flash e até mesmo gravar valores maliciosos neles.

**Obtendo Dados e Firmware do Dispositivo**

Se outras técnicas de obtenção do firmware falharem, o JTAG é nossa opção alternativa. Podemos usar o JTAG com o comando dump\_image para despejar o firmware do sistema de arquivos.

Isso é mostrado aqui, onde despejamos o firmware do mesmo chip flash indicando o endereço de onde queremos despejar o conteúdo e a quantidade de dados que queremos despejar.

Uma imagem contendo Texto

Descrição gerada automaticamente

**Leitura de Dados do Dispositivo**

Também podemos ler dados seletivamente de endereços de memória específicos usando JTAG. Isso é útil quando sabemos o endereço exato que queremos ler e, posteriormente, talvez modificar.

Podemos usar o comando mdw seguido do endereço e do número de blocos a serem lidos.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Conforme mencionado anteriormente, este firmware contém uma função de autenticação para acesso UART. A senha neste caso é armazenada em um deslocamento de d240.Dado que sabemos o endereço base do armazenamento flash – 0x08000000 – nóspode usar o comando mdw para despejar a senha do endereço base +endereço que é 0x0800d240, conforme mostrado aqui.

Texto

Descrição gerada automaticamente

Ao converter o valor 61 74 74 69 66 79 da saída, que está em hexadecimal para ASCII, obtemos a senha real, que neste caso é attify. Da mesma forma, também podemos gravar um novo valor no endereço de memória para alterar a senha usada pelo dispositivo para autenticação UART.

Esta é uma demonstração simples de como você pode usar a leitura de conteúdo da memória a seu favor durante a exploração. Ao realizar isso durante seus testes de penetração, certifique-se de procurar por qualquer coisa de interesse potencial, então descubra o endereço no hexdump ou através de um disassembler e, finalmente, leia e escreva valores usando a depuração JTAG.

## **Depuração via JTAG com GDB**

Frequentemente, você precisa depurar binários e firmware via JTAG para entender a funcionalidade de uma maneira muito melhor e modificar alguns valores de registradores ou conjuntos de instruções e alterar o fluxo de execução do programa.

Agora que já estamos familiarizados com a depuração usando GDB, usaremos o GDB para depurar um binário que flashamos via JTAG. O binário pode ser baixado dos exemplos de código do livro disponíveis em [invalid URL removed]> (endereço omitido por segurança). Depois de termos flasheado o binário, prosseguimos com a depuração do tempo de execução do binário com JTAG e GDB.

Agora você deve estar se perguntando como podemos conectar o GDB ao processo de destino através do JTAG. A resposta é que sempre que executamos o OpenOCD para um destino para realizar a depuração JTAG, ele também habilita dois serviços diferentes, o primeiro sendo telnet pela porta 4444, que usamos para interagir com o OpenOCD, e o outro sendo GDB pela porta 3333, que podemos usar para depurar binários em execução no dispositivo de destino.

Para fazer isso, inicie o GDB-Multiarch e forneça o binário que queremos depurar, que neste caso é o binário de firmware.bin, chamado authentication.elf. Uma vez conectado, também definiremos a arquitetura como arm e a apontaremos para a porta 3333, onde o OpenOCD conectou o gdbserver com o processo em execução.

Texto

Descrição gerada automaticamente

Uma vez que você tenha o GDB para arm instalado e funcionando, você pode prosseguir e configurar pontos de interrupção usando hbreak ou break para melhor analisar o binário e analisar toda a pilha e registradores quando o ponto de interrupção for atingido. hbreak é usado para definir um ponto de interrupção assistido por hardware. break pode ser usado para definir um ponto de interrupção normal em uma instrução, local da memória ou função.

A primeira coisa que queremos fazer é olhar para as funções neste binário. Para fazer isso, usaremos o comando info functions mostrado aqui.

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Como podemos ver, o verifypass é uma função para comparar a senha fornecida pelo usuário com a senha real usando a instrução strcmp no endereço 0x080002e4. Com base no resultado, se a autenticação for concedida, ela então ramifica para 0x8000290 para abrir a porta ou continua se as senhas não coincidirem.

Para descobrir a senha real, podemos definir um ponto de interrupção na instrução strcmp e analisar os registradores r0 e r1, que conterão os dois valores sendo comparados. Um desses valores será a senha fornecida pelo usuário e o outro valor será a senha real.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Depois de definir o ponto de interrupção, você pode digitar "c" para continuar a execução do programa. O próximo passo é se conectar pela UART e fornecer uma entrada, para que a função verifypass seja chamada e nosso ponto de interrupção seja atingido. Para conectar ao mesmo dispositivo alvo pela UART, usaremos os pinos A2 e A3, que são Tx e Rx do STM32, e os conectaremos aos pinos D1 e Crachá de Afirmação D0. A Figura 6-13 mostra como seria a conexão final.

Diagrama

Descrição gerada automaticamente

Nós nos conectamos ao console UART em outro terminal e usamos o comando screen para conectar a /dev/ttyUSB0 com uma taxa de transmissão de 9600 bauds.

Texto

Descrição gerada automaticamente

Vamos digitar "testing" como senha aqui e pressionar Enter. Assim que pressionarmos Enter, podemos ver que o ponto de interrupção foi atingido na sessão GDB, conforme mostrado aqui.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

Texto

Descrição gerada automaticamente

Nesse ponto, podemos analisar os registradores usando os registradores de informação.

Tabela

Descrição gerada automaticamente

Vamos ver o que há em r0 e r1 usando o comando x/s, que examina o valor como uma string.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Você pode ver que r0 contém a senha inserida, que é testando e r1 contém a senha real, que é attify. Aqui podemos alterar o valor de r0 e configurá-lo para attify e digitar c para continuar a execução.



Agora você pode ver em nosso terminal de tela que a autenticação foi concedida, conforme mostrado na Figura 6-14.

Texto

Descrição gerada automaticamente com confiança média

É assim que podemos explorar um binário pelo JTAG, realizar depuração em tempo real do binário e modificar um dos registradores para tornar a autenticação válida.

**Conclusão**

Neste capítulo, demos uma olhada em uma das maneiras mais interessantes de explorar dispositivos embarcados, que é através do JTAG. No entanto, as técnicas e o conteúdo abordados neste capítulo têm como objetivo ajudá-lo a começar com a depuração JTAG e espero que você aplique essas habilidades em dispositivos do mundo real para ir ainda mais longe.

Além disso, uma vez que você tenha obtido acesso de depuração JTAG, depende de até onde você deseja ir. Isso significa que, como você pode depurar binários, pode encontrar mais vulnerabilidades nos vários binários em execução no dispositivo, o que pode ajudá-lo a comprometer o dispositivo de outras maneiras.