UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Departamento de Computação

Eric Pereira Queiroz Moreira

ATAQUES DE FIRMWARE NA CADEIA DE PRODUÇÃO

São Carlos - SP

2023

Eric Pereira Queiroz Moreira

ATAQUES DE FIRMWARE NA CADEIA DE PRODUÇÃO

Trabalho de Conclusão do Curso de Bacharelado em Ciência de Computação.

Orientação Prof. Dr. Paulo Matias

São Carlos - SP

2023

Dedico este trabalho de pesquisa aos meus pais, as duas pessoas que mais amo e admiro nesse mundo. Espero que este seja mais um degrau na escada do sucesso que vocês me ensinaram a trilhar. Seu apoio me serviu como mola propulsora mesmo durante os momentos mais difíceis, e por isso serei grato para todo o sempre.

Agradecimentos

Agradeço primeiramente à Deus por todas as bençãos oferecidas, mesmo que eu tenha demorado para percebê-las. Em retrospectiva, vejo que Ele foi muito bondoso comigo.

Aos meus pais, pelo amor, incentivo e apoio incondicional. Esse trabalho pertence a eles tanto quanto pertence a mim, pois eles foram meu suporte durante todo esse tempo. Minha mãe que sempre me motivou a seguir em frente e nunca desistir mesmo sob desânimo, minha heroína. E meu pai, que sempre apoiou a família de forma sólida como rocha mas com um coração bondoso e acalentador.

Aos meus amigos que conheci nas diversas jornadas da vida. Meus amigos de São Carlos, Campinas, Vitória, Rio de Janeiro, São José dos Campos e, especialmente, meus amigos de longa data de Belém. Obrigado por sua compreensão comigo e por sempre estarem do meu lado nos tempos bons e ruins. Não sei o que seria de mim sem nossos momentos de lazer como festas e/ou jogatinas, nossas conversas nos momentos mais importantes ou nossos simples almoços juntos. Farei meu melhor para compensar a assistência inabalável de vocês.

Ao prof. Dr. Paulo Matias, pela orientação acadêmica, apoio e confiança. Ele se provou um ótimo orientador e uma fonte inesgotável de conhecimento.

Ao Mateus Gava e a equipe técnica da SIn (Secretaria de Informática) UFSCar por ceder equipamentos, espaço e dicas valiosas que se provaram essenciais para a conclusão do trabalho.

Aos colegas Rodrigo Laneth, Davidson Francis e Rodrigo Rubira Branco pela troca de ideias que contribuiu imensamente com o resultado final.

À Universidade Federal de São Carlos, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior. Nesse solo criei lembranças muito boas que levarei para o resto da vida.

"O aspecto mais triste da vida de hoje é que a ciência ganha em conhecimento mais rapidamente que a sociedade em sabedoria."

(Isaac Asimov)

Resumo

Esse trabalho trata do procedimento de identificar e corromper firmware para atacar uma cadeia de produção. Nele, explica-se o que é uma cadeia de produção e qual a relevância e impacto de atacá-la. Em seguida, estabelece-se um ambiente de teste e realiza-se o estudo de diversos componentes da placa mãe visando encontrar um vetor de ataque. O trabalho então determina um vetor e realiza diversos experimentos de forma a ilustrar como um fornecedor malicioso poderia vir a infectar o chip, mostrando também o resultado final quando o chip é restabelecido no ambiente de testes do projeto.

Palavras-chave: Cadeia de produção, firmware, chip, segurança, engenharia reversa, ataque.

Abstract

This work is about the process of identifying and corrupting a piece of firmware to attack a supply chain. In it, it is explained what is a supply chain and the relevance and impact of studying its security. Then, it establishes a test environemnt and studies several motherboard components in order to pinpoint one which could be revelant as an attack vector. After a vector is found, the research then takes apart and infects the selected chip in order to do a proof of concept on how this chip could end up being compromised by a malicious supplier. Finally, it reintroduces the chip in the test environment and shows the end result.

Keywords: Supply chain, firmware, chip, security, reverse engineering, attack, red team.

Lista de ilustrações

Sumário

1 Introdução

1.1 Segurança da Cadeia de Produção

1.1.1 Conceito e Práticas

A segurança da cadeia de produção (conhecida como *supply chain*) refere-se ao gerenciamento e mitigação de riscos de operações envolvendo as diferentes etapas da cadeia. Isso pode incluir tanto logística, transporte, manufatura, agentes externos, etc.

Nesse caso, o objetivo é identificar, avaliar e mitigar riscos ou pontos sensíveis envolvendo agentes externos. Isso se dá tanto no âmbito da segurança física para produtos e processos quanto no âmbito da segurança cibernética para *softwares* e serviços.

Além disso, outros pontos relevantes são a integridade do produto, ou seja, garantia de que os produtos não são adulterados, danificados e/ou falsificados durante manufatura e transporte, e o controle de qualidade, que garante que exista verificação e certificação de que os produtos atinjam o padrão de qualidade exigido.

Nesse mesmo raciocínio, quando se fala de segurança da cadeia de produção também destaca-se bastante gerenciamento e mitigação de erros em eventuais crises. Denomina-se resiliência quando uma cadeia possui planos de contingência para lidar com interrupções inesperadas. Além disso, também avalia-se o fator de *compliance*, que indica conformidade com regulações e leis aplicáveis.

Por fim, outro fator relevante para segurança de cadeia é a rastreabilidade: estabelecer formas para rastrear produtos ao longo de todo seu ciclo de vida dentro da cadeia, facilitando assim também identificação de erros e medidas de correção.

1.1.2 Contexto e Relevância

A relevância do tema provém do aumento recente de ataques envolvendo segurança de *supply chain*. Nos últimos cinco anos, dois casos graves com potencial de expor dados de milhões de usuários e servidores web se destacaram. A diferença de tempo é destacadamente pequena e mostra uma necessidade de estudar o funcionamento e possível mitigação desse tipo de ataque.

O primeiro exemplo trata-se do *Sunburst* (ou Supernova) (RAMAKRISHNA, 2021). Esse ciberataque ocorreu em Março de 2020 e teve como alvo a *SolarWinds*, uma empresa texana que provinha *software* de monitoramento de redes para o governo estadunidense e várias empresas renomadas do ramo.

Atacando uma provedora de uma cadeia sensível com um código vulnerável, todos os membros da cadeia que utilizavam o *software* da *SolarWinds* tiveram seus dados expostos. Esse ataque foi considerado um dos piores incidentes de ciberespionagem sofridos pelos Estados Unidos, devido à sensitividade dos alvos e à longa duração (alegadamente entre oito e nove meses) na qual os *hackers* tiveram acesso aos dados.

Estima-se que cerca de 200 organizações ao redor do mundo foram afetadas pelo *Sunburst*, e algumas dessas consequentemente sofreram falhas de segurança de dados, dentre elas os governos dos Estados Unidos e Reino Unido, Microsoft, dentre outras.

O segundo exemplo ocorreu no dia 28/03/2020, onde dois *commits* maliciosos foram feitos no repositório *php-src*, comprometidos pelo servidor git.php.net, e não por causa de uma conta individual. Esse ataque ocorreu devido ao fato da organização PHP manter uma infraestrutura Git própria, que não seguia os padrões de segurança estabelecidos e fornecidos para os usuários por ferramentas como *Github* e *Gitlab*, tornando-os vulneráveis a ataques que seriam mitigados em tais ambientes.

Desde então, os repositórios do *Github*, os quais previamente serviam apenas como repositórios espelhados dos originais, passaram a ser canônicos. O ataque nesse caso foi contido antes de ter sido publicado e distribuído, mas é importante verificar como essa falha de segurança ainda foi crítica. Caso ele não tivesse sido capturado, teria afetado todos os membros posteriores da cadeia: toda *codebase* que utiliza PHP teria sido afetada e comprometida pelo código malicioso.

Ambos os exemplos apontam como usuários maliciosos estão cada vez mais dispostos a observar e atacar elementos vulneráveis de uma cadeia de produção envolvendo desenvolvimento de software com o objetivo de atingir softwares de maior porte que utilizem esses softwares vulneráveis. Cadeias de suprimento estão cada vez mais comuns no âmbito de software e elas podem escalar pequenas falhas em danos de grande porte.

1.2 Cadeia de Produção em Hardware

1.2.1 Cadeia de Autenticação em Hardware

Uma cadeia de autenticação nesse contexto é um modelo para operações seguras num sistema computacional. A ideia é que cada componente autentique o próximo (representando a "cadeia") e dificultando ataques de malware no hardware. Dessa forma, ao invés de todos os componentes se autenticarem independentemente durante a inicialização, cada um deles participa da autenticação do próximo.

Um detalhe importante é pensar em qual componente é o primeiro da fila, a "raiz" da cadeia. Essa raiz deve conter as chaves utilizadas nas funções criptográficas do sistema e ser responsável pelo processo de *secure boot* como um todo, pois ela o inicia. Como

nenhum componente a autentica, sendo ela inerentemente confiada, ela deve ser segura por padrão. Em geral, ela é um módulo de segurança a parte no firmware ou implementada como um componente dentro de um processador ou sistema maior.

Além disso, esses componentes de autenticação podem ser de função fixa ou programável. Uma raiz de autenticação de função fixa é controlada pelo firmware. Em geral, são compactas e executam um conjunto específico de funções como gerenciamento de chaves, encriptação de dados e validação de certificados.

Já uma cadeia de autenticação de função programável é construída ao redor de uma CPU. Além das funções iniciais já mencionadas, ela possui um conjunto mais complexo de funções relacionadas à segurança do sistema. Em geral, os benefícios desse tipo são a versatilidade e a possibilidade de atualizações.

Dessa forma, num campo dinâmico onde novos ataques e possibilidades são desenvolvidos numa frequência considerável, componentes de função programável podem se manter seguros.

Atualmente, o componente mais utilizado como raiz da cadeia de autenticação é o TPM (Trusted Platform Module, traduzido como Módulo de Plataforma Confiável). Ele é um chip separado da placa-mãe e é de função fixa.

Como mostrado na Figura 1, o módulo TPM faz todo o trabalho de gerenciar chaves utilizadas para autenticação e para encriptar mensagens trocadas entre os módulos. Quando a inicialização do sistema é feita, cada módulo (BIOS Boot Block, BIOS, OS Loader, etc.) autentica o seguinte e isso compõe a cadeia.

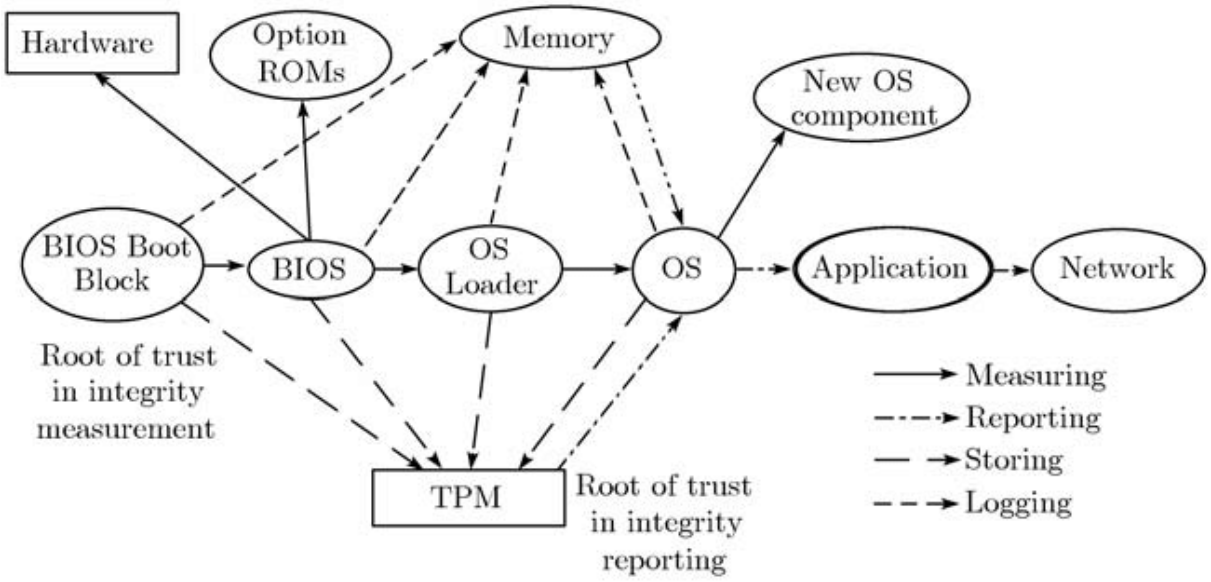


Figura 1 – Cadeia de Segurança em Hardware utilizando o módulo TPM como raiz, como especificado no TCG (Trusted Computing Group). Fonte: Shen et al. (2010).

Esse formato é descrito pelo TCG e é considerado a especificação padrão da

indústria de produção de hardware (SHEN et al., 2010).

1.2.2 Fornecedor Malicioso

Após analisar o que é uma cadeia de produção, sua relevância e como o conceito se aplica num contexto computacional, mais especificamente no âmbito do hardware, deve-se analisar como funcionará a classe do ataque a ser analisado nesse trabalho.

O contexto a ser estudado aqui é o de ataques de segurança cibernética onde o atacante explora vulnerabilidades na cadeia de produção para comprometer os dados e/ou sistemas do alvo. Nesse cenário, o fornecedor/vendedor atua como um agente que entrega produtos, serviços ou softwares para a organização alvo. Esses produtos serão propositalmente comprometidos de forma a gerar uma vulnerabilidade que será posteriormente explorada.

Existem diversas formas de se realizar tal tarefa. Como observado no caso da *SolarWinds* (RAMAKRISHNA, 2021), pode-se comprometer a cadeia de produção, injetando código malicioso ou um *backdoor* via um software que posteriormente será utilizado por várias organizações.

Outra forma bastante comum é utilizar seus privilégios como fornecedor para acessar dados sensíveis da organização e extraviá-los para propósitos variados, como vender os dados em sites com tal propósito ou utilizá-los para obter uma vantagem competitiva em relação à organização alvo.

Além disso, caso o fornecedor malicioso forneça componentes de firmware para a empresa, ele pode manipular o código embarcado no componente para propositalmente gerar vulnerabilidades e, assim, comprometer o sistema do alvo.

Entretanto, o caso que será analisado mais a fundo será o conceito de injeção de vulnerabilidades e/ou backdoors por parte de um fornecedor de hardware ou componentes de hardware. Esses ataques podem tomar vários formatos, injetando vulnerabilidades via circuitos integrados, conectores extras, modificações no firmware, etc. Da mesma forma, o componente de hardware infectado pode estar em diversos níveis: placa-mãe, microprocessador, até mesmo dispositivos periféricos.

Uma vantagem clara é que esse tipo de ataque, em geral, faz-se mais difícil de detectar do que um ataque baseado em software. Métodos clássicos de defesa como monitoramento de redes e anti-vírus diversos são ineficazes nesse cenário. Além disso, hardware malicioso injetado não deixa modificações na memória ou arquivos do sistema, fazendo com que seja um vetor de ataque bastante eficaz.

Obviamente, ainda assim existem contra-medidas que devem ser levadas em conta e lidadas pelo atacante. Empresas que importam hardware de terceiros podem ter medidas

de segurança, testar funcionalidades do hardware, realizar teste de penetração para garantir a integridade dos componentes, dentre outras rotinas de garantia e controle de qualidade.

1.3 Objetivos

O objetivo desse projeto é utilizar uma prova de conceito para demonstrar um ataque na cadeia de produção, mostrando mais especificamente brechas que podem ser encontradas por um fornecedor malicioso com acesso ao firmware.

Dessa forma, pode-se pesquisar possíveis ataques e métodos que possam ser utilizados para injetar código malicioso num ambiente, por exemplo, de um *data center*.

Como o trabalho se trata de uma invasão, obviamente não é possível e prudente utilizar um alvo real. Para isso, deve-se realizar a montagem de um ambiente de teste fechado e contido. Ademais, como um ataque desse tipo é potencialmente destrutivo, o ambiente deve ser de recursos não-essenciais, que possam vir a ser descartados após sua utilização.

1.4 Organização do Trabalho

O trabalho está dividido em quatro seções principais: no capítulo 1, introduz-se a ideia por trás do projeto, as motivações para estudar a área do conhecimento abordada nele e uma breve introdução dos conceitos necessários para compreender o assunto tratado.

Já no segundo capítulo, trata-se das metodologias teóricas por trás do trabalho. Mostra-se o desenvolvimento de um ambiente de testes, porque usá-lo e suas especificações. Dado o ambiente, descreve-se também as técnicas de invasão utilizadas e suas respectivas eficácias. Para isso, usa-se literaturas como referência teórica para identificar a qual modelo o experimento deverá se adequar. É importante analisar como o ataque deverá se comparar para verificação posterior durante o experimento.

No terceiro capítulo, descreve-se o experimento realizado em detalhe, como as técnicas de invasão foram aplicadas no ambiente de testes utilizado, relata-se o ocorrido e, por fim, discorre-se sobre o esperado teoricamente e o que realmente ocorreu na prática, para assim comparar as diversas técnicas analisadas.

Por fim, no quarto capítulo é tratada a conclusão e as considerações finais do trabalho. Dado o resultado do trabalho, são analisados quais objetivos foram concluídos, quais teses foram comprovadas ou não dado o experimento realizado. Além disso, discute-se também aspectos faltantes do trabalho com possíveis ideias futuras de para onde seguir numa linha de pesquisa derivada deste.

2 Modelo Teórico

2.1 Ambiente de Testes

2.1.1 Servidor Utilizado

O primeiro passo para montagem de um ambiente de testes seria utilizar uma máquina para testes e mudanças gerais em seu *firmware*. Por via de doações de material de uma empresa parceira do laboratório (por meio de projeto de extensão) no qual o experimento foi realizado, escolheu-se para esse propósito um servidor Rack Mount Chassis PowerEdge R710 Dell Inc. 9MCS9S1 64 bits, cujo chassis é exposto na Figura 2. A máquina foi considerada material passível de descarte, pois como citado anteriormente, testes potencialmente destrutivos foram realizados na máquina.

Figura 2 – Vistas frontal e traseira do Servidor Dell R710 PowerEdge exposto, sem a peça de proteção externa. Fonte: Dell (2022).

Também é importante ressaltar que a máquina foi mantida em ambiente para uso próprio no laboratório do projeto dentro da universidade. Ela não foi utilizada para nenhum outro fim que não fosse relevante ao projeto, dessa forma eliminando qualquer

possível influência no seu comportamento. A ideia é, obviamente, simular uma máquina que seria infectada durante seu fornecimento. Dessa forma, qualquer utilização prévia subtrairia do cenário ao qual ela serve.

A máquina contou com duas CPUs modelo Intel(R) Xeon(R) CPU X5670 com 18 (dezoito) unidades de memória RAM DIMM DDR3 Synchronous Registered (Buffered) 1333 MHz de 4 GB cada. Além disso, também contou-se com uma placa controladora de rede do modelo Ethernet Interface NetXtreme II BCM5709 Gigabit Ethernet.

Por fim, o ponto mais crucial para análise foi a placa-mãe do servidor, uma Dell Inc. modelo 00NH4P versão A12, que pode ser observada na Figura 3. A placa mãe em si é um modelo complexo e cheio de partes que necessitam ser analisadas com mais precisão.

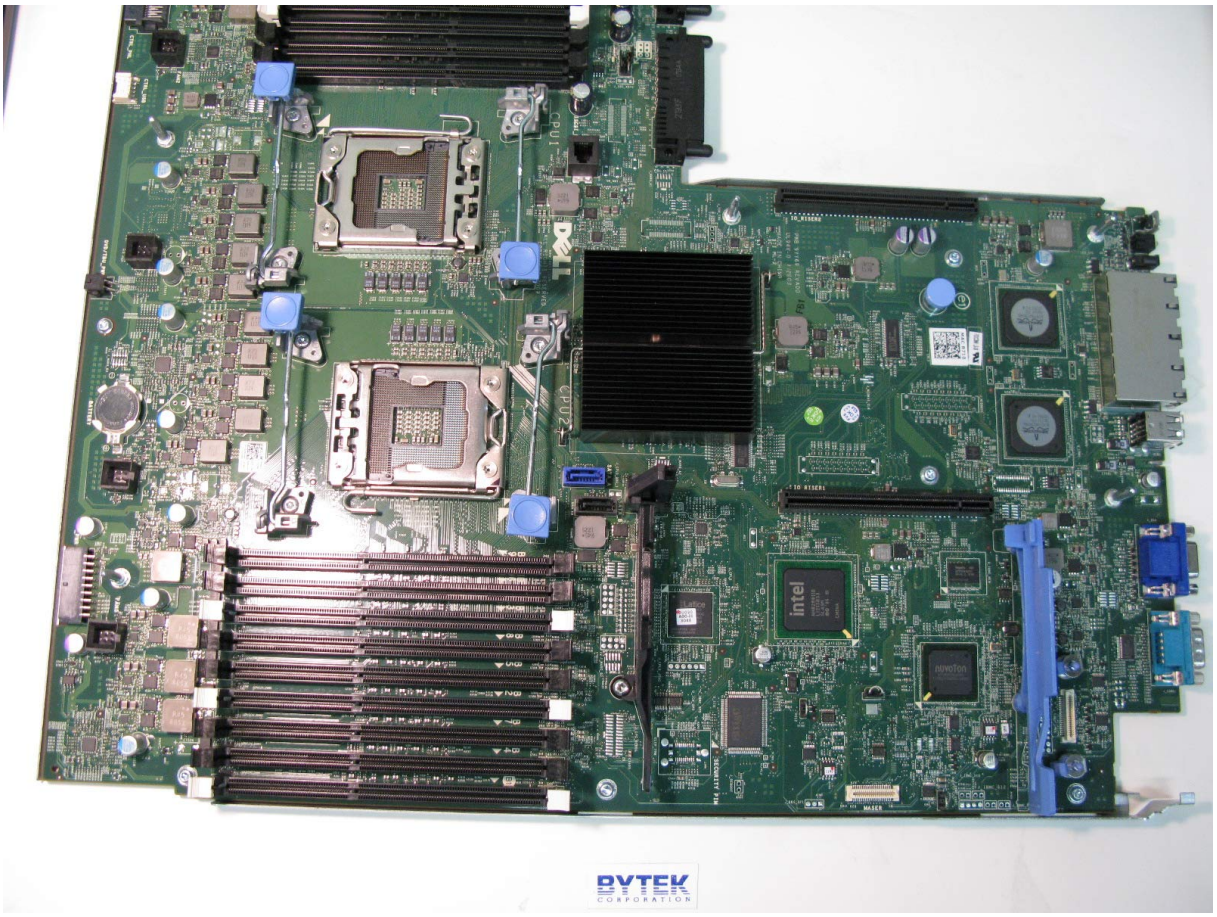


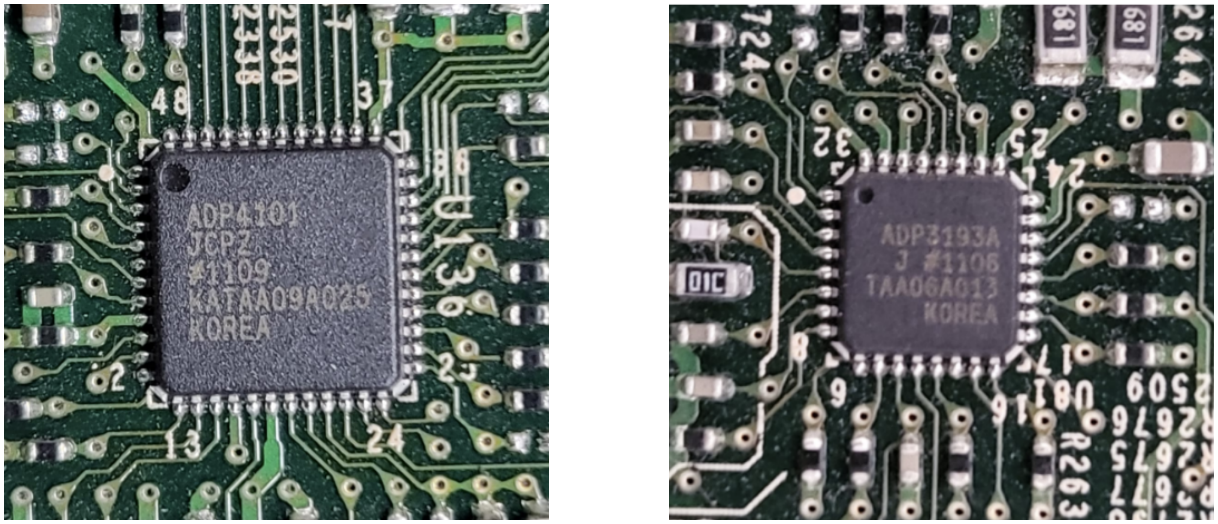
Figura 3 – Visão completa da placa mãe 00NH4P utilizada no projeto. Fonte: BYTEK (2023).

Inicialmente, a ideia seria utilizar os esquemáticos da placa-mãe para identificar os principais vetores clássicos de ataque (BIOS, BMC, placa de rede, etc.). Entretanto, como trata-se de um *hardware* proprietário da empresa Dell Inc., tais informações não são publicamente divulgadas para proteção de sua propriedade intelectual. Inclusive, tentou-se contato com a própria equipe da Dell Inc. para fornecimento de mais detalhes do produto, porém infelizmente o suporte para o Dell PowerEdge R710 foi descontinuado em 2017.

Dessa forma, fez-se necessário construir um catálogo manual para identificação de quase todos os chips encontrados na placa-mãe, em especial as mais relevantes citados acima.

2.1.2 A Placa 00NH4P

Dentro do catálogo da placa-mãe, destacam-se diversos chips e peças de *firmware* que podem vir a ser potenciais vetores de ataque, como mencionado anteriormente. Nesta seção, documentou-se todos os componentes principais da placa **00NH4P** e sua função. Posteriormente serão analisados possíveis vetores de ataque relacionados a alguns desses.



(a) Chip ADP4101, que atua como AFE na placa.

(b) Chip ADP3193A, que atua como regulador e controlador síncrono de conversor *buck*.

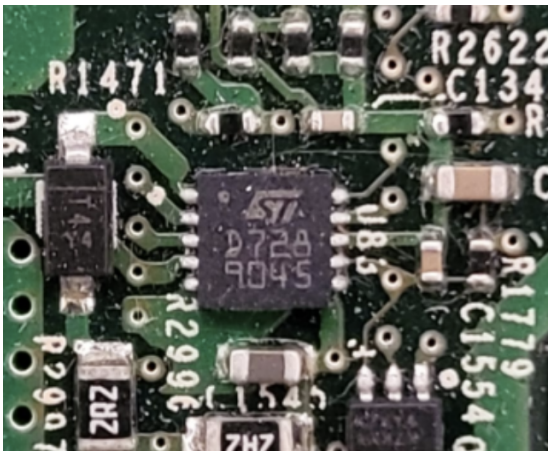
Figura 4 – Imagens dos chips 1 e 2 analisados

Esse primeiro chip encontrando na Figura 4, modelo **ADP4101** (ON SEMI, 2023), é um AFE (Analog Front-End). Ele é responsável por condicionar, amplificar, filtrar e processar sinais analógicos antes que eles sejam convertidos para formato digital para processamento posterior. Geralmente, um AFE é projetado para atender a um tipo específico de entrada analógica, como sinais de áudio, sinais de sensores de temperatura, pressão, etc.

Já o segundo chip da mesma Figura, modelo **ADP3193A** (ANALOG DEVICES, 2023), atua como regulador e controlador síncrono de conversor *buck*. Ou seja, opera como um controlador para um circuito conversor *buck*, mantendo a operação sincronizada e eficiente para controlar a tensão de saída de acordo com as necessidades do sistema.

O primeiro chip encontrando na Figura 5, modelo **8643AE** (MAXIM INTEGRATED, 2023), é um conversor DC-DC. Ele converte fonte de energia de corrente contínua de um nível de tensão para outro. Dito isso, o segundo chip encontrado na mesma Figura,

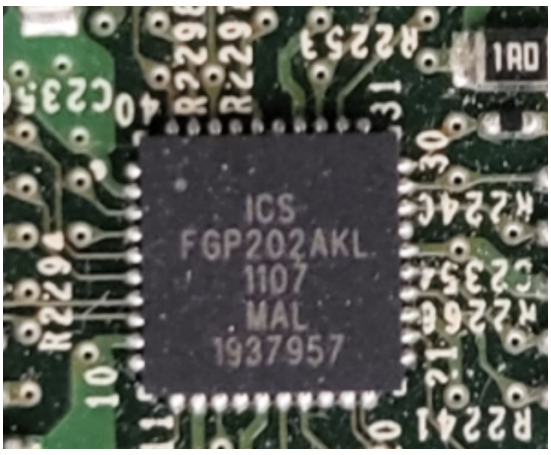
(a) Chip 8643AE, que atua como conversor DC-DC na placa.



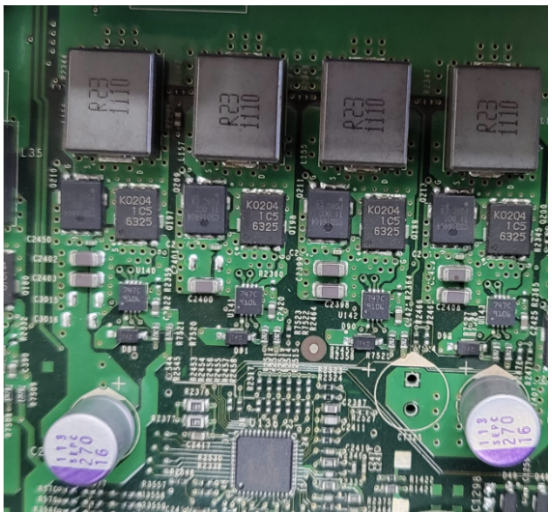
(b) Chip modelo ST D728, possivelmente um regulador de tensão.

Figura 5 – Imagens dos chips 3 e 4 analisados

com marcação **ST D728**, não foi identificado com plena certeza, porém foi localizado um anúncio de venda que o classificava como regulador de tensão (JU FAN STORE, 2023).



(a) Chip da família FGP202AKL que atua como *clock* periférico.

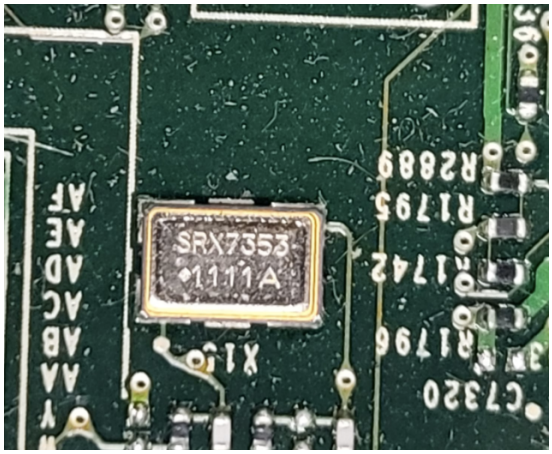


(b) Chip modelo K0204, possivelmente um regulador de tensão.

Figura 6 – Imagens dos chips 5 e 6 analisados

Assim como nas figuras anteriores, vê-se na Figura 6 dois chips. O da subfigura (a) é um **FGP202AKL** (IDT, 2023), um *clock* periférico. Ou seja, ele atua como a frequência do sinal de relógio que é fornecida aos periféricos, como timers, comunicações seriais (UART, SPI, I2C), conversores analógico-digitais (ADC), e outros módulos que requerem sincronização para funcionar corretamente. Já na Figura (b) consegue-se visualizar quatro unidades de chip **K0204**, que não foi identificado com plena certeza, porém foi localizado um anúncio de venda que o classificava como regulador de tensão (SHOP233861, 2023).

Pode-se ver na Figura 7 dois chips. O da subfigura (a) é o **SRX7353** (DIODES



(a) Cristal cerâmico SRX7353. Vários componentes similares foram encontrados na placa-mãe.

(b) Chip da família K7KY4 que atua como conversor DC-DC.

Figura 7 – Imagens dos chips 7 e 8 analisados

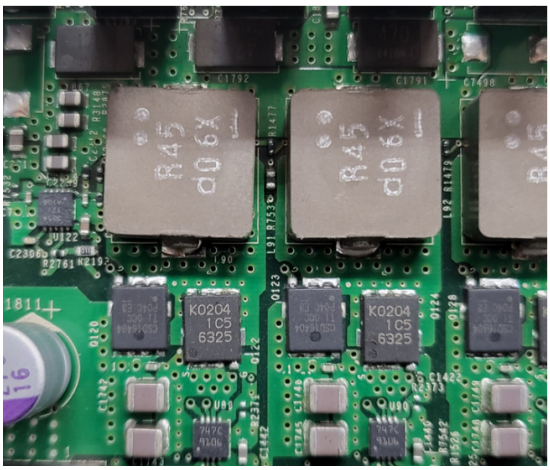
INCORPORATED, 2023), que é um cristal cerâmico, geralmente utilizado como parte de um circuito oscilador. Já o chip da subfigura (b) trata-se de um chip **K7KY4**, que também atua como conversor DC-DC.

(a) Chip principal da Placa de rede Broadcom BCM5709CC0KPBG.

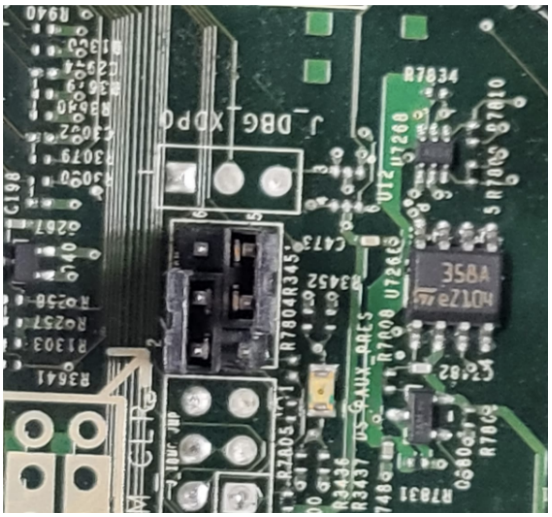
(b) Chip AT45DB041D, flash SPI de 4 Mbit.

Figura 8 – Imagens dos chips 9 e 10 analisados

Agora, na Figura 8 vê-se uma parte importante da placa, ainda mais no contexto de ataques em *firmware*, a interface de rede. Na Figura (a), vê-se o chip com toda a lógica necessária para seu funcionamento (BROADCOM, 2023), sendo que a placa mãe possui duas unidades desse chip. Já na Figura (b), vê-se um chip **AT45DB041D** (ATMEL, 2023) que funciona como uma memória Flash SPI de 4 MBit, que pode potencialmente conter algum tipo de *firmware*.



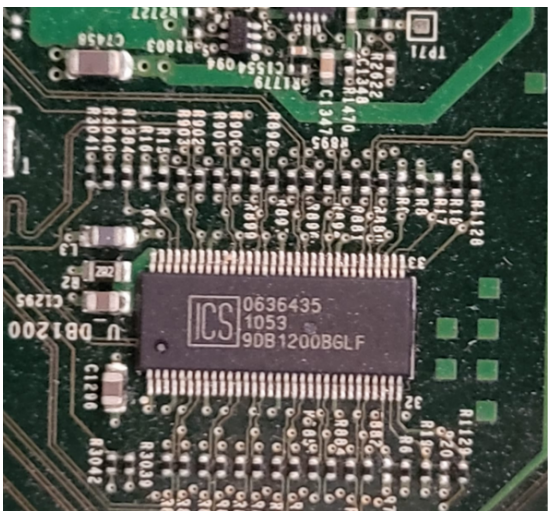
(a) Além de conter chips K0204 já analisados previamente, também encontra-se chips modelo 747C.



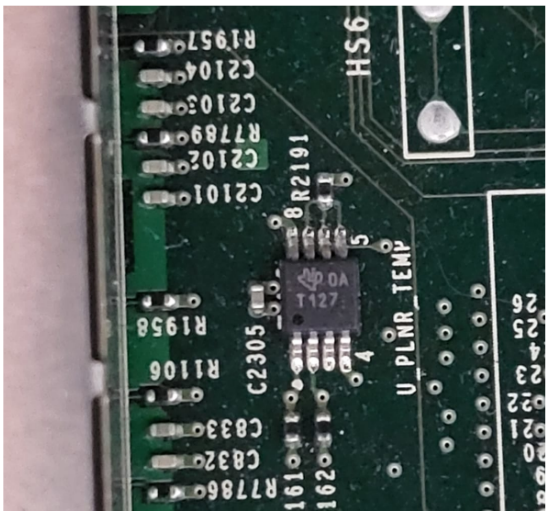
(b) Chip LM358A que atua como um conjunto de 2 (dois) amplificadores operacionais com compensação de frequência.

Figura 9 – Imagens dos chips 11 e 12 analisados

Dentro da Figura 9, a subfigura (a) mostra chips **747C**, que não foram identificados. Além disso, a Figura (b) mostra um **LM358A** (NXP, 2023), um conjunto de dois amplificadores operacionais com compensação de frequência. Esse componente emite uma ampla faixa de frequências de forma majoritariamente estável, minimizando variações indesejadas no amplificador.



(a) Chip modelo ICS 9DB1200BGLF, que atua como dispositivo de *buffer* diferencial.

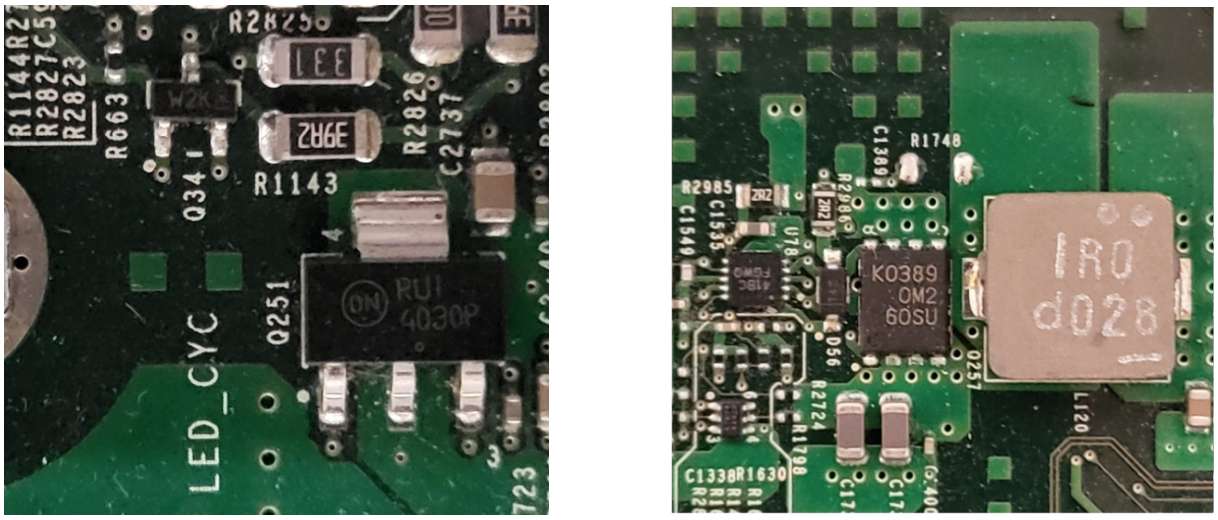


(b) Chip T127, um sensor de temperatura.

Figura 10 – Imagens dos chips 13 e 14 analisados

Agora, como é possível ver na Figura 10, tem-se um chip **ICS 9DB1200BGLF** (JOTRIN, 2023), um dispositivo de *buffer* diferencial. Esse *buffer* provê 12 (doze) *clocks* diferentes em frequências de 100 a 400 MHz. Já o chip **T127** (TEXAS INSTRUMENTS,

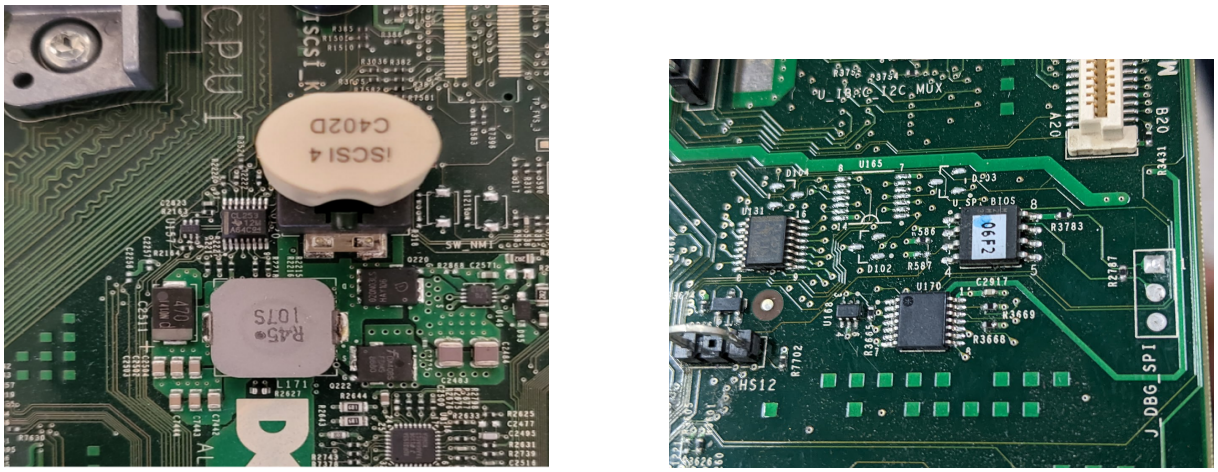
2020) é um sensor de temperatura.



(a) 4030P, um transistor PNP. (b) K0389, um MOSFET.

Figura 11 – Imagens dos chips 15 e 16 analisados

No contexto da Figura 11, o primeiro componente (Mostrado na subfigura (a)) é do modelo **4030P** (ON SEMI, 2021), um transistor PNP capaz de trabalhar a até 3 amperes e 40 volts. Já o **K0389** (RENESAS, 2023b) é um Transistor MOSFET de potência, que é utilizado primariamente como interruptor eletrônico no circuito.



(a) Chips de modelo CL253 e 020N03LS. (b) Chip de modelo MX25L3206E (com adesivo 06F2), uma memória flash NOR.

Figura 12 – Imagens dos chips 17 e 18 analisados.

Agora, observando a Figura 12, observa-se na subfigura (a) um chip da família **CL253**, que atua como multiplexador FET Dual de tensão baixa (TEXAS INSTRUMENTS, 2018), ou seja, é capaz de realizar funções de multiplexação (mux) e demultiplexação (demux) usando transistores de efeito de campo (FETs) em uma configuração dual (dupla). Já o chip **020N03LS** (INFINEON, 2023) é outro transistor MOSFET.

Já o chip da subfigura (b), marcado com o adesivo 06F2, trata-se de um **MX25L-3206E** (MACRONIX, 2023). Ele é um dispositivo de memória não volátil usado para armazenamento de dados em sistemas eletrônicos que necessitam de armazenamento de dados de forma permanente mesmo quando a energia é desligada. Esse chip é extremamente importante como vetor de ataque e será posteriormente explorado.

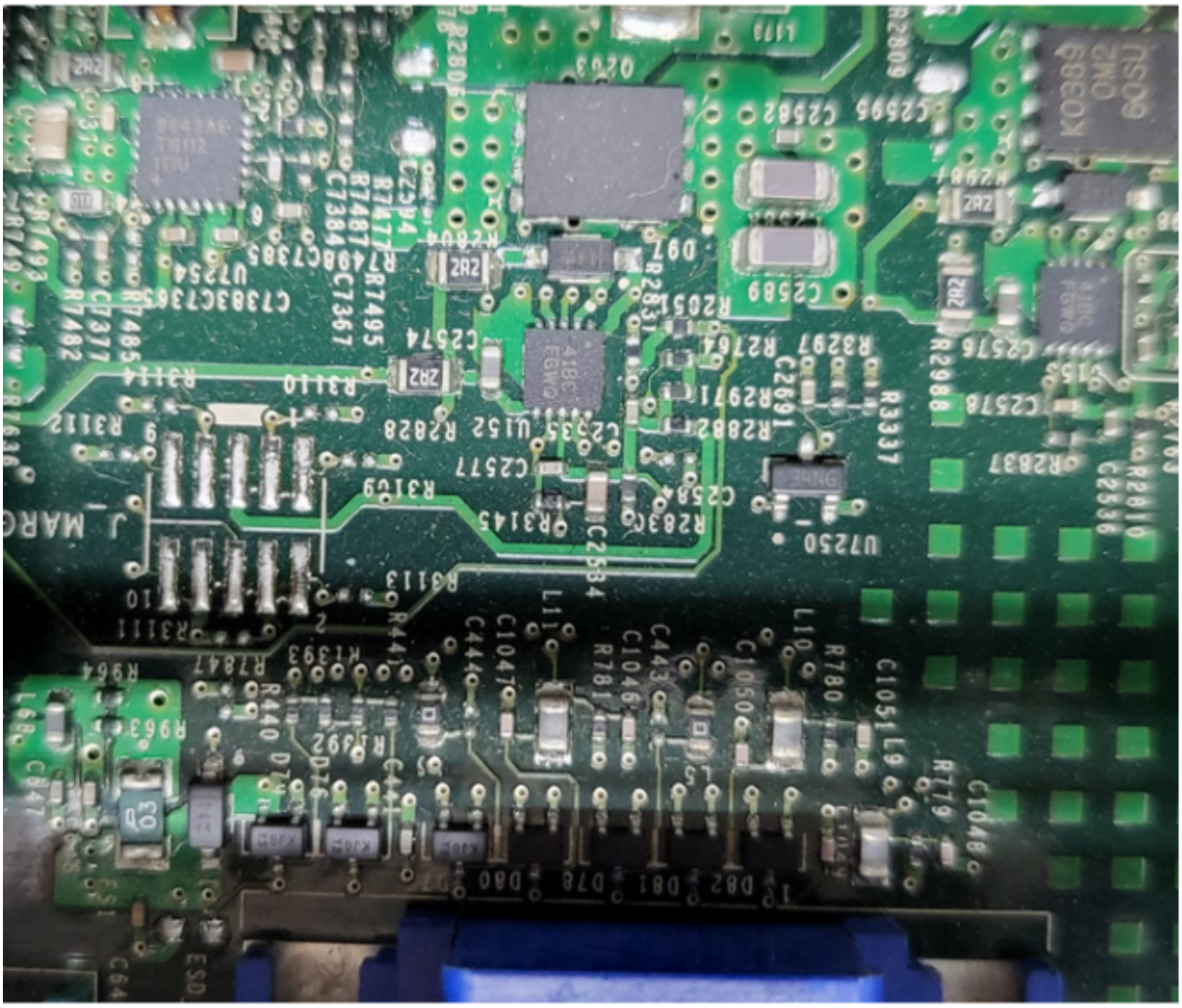


Figura 13 – Dois chips 41BC, um K0389 e um chip 8643AE.

Na Figura 13, vê-se quatro chips. Dois chips são marcados como **41BC** (RENESAS, 2023a), ou seja, fazem parte de uma família específica que compõe um esquema completo de proteção e controle de conversores DC-DC. Já o chip **K0389** é também um transistor de efeito de campo. Por fim, o **8643AE** é também um conversor DC-DC, provavelmente associado aos chips 41BC.

2.2 Vetores de Ataque

Dado os componentes de *hardware* da placa 00NH4P, discutidos na seção anterior, pode-se analisar aqueles que contém *firmware* que possa ser modificado a fim de gerar

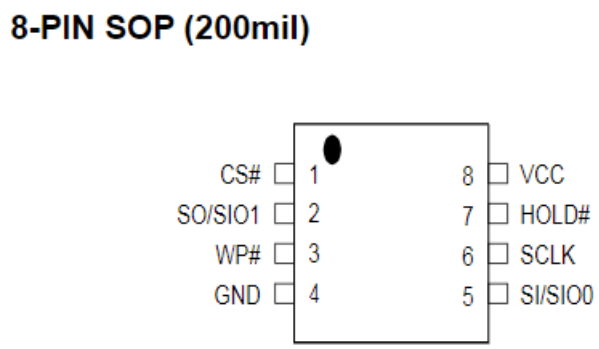


Figura 14 – Pinagem do chip analisado de acordo com seu datasheet.

possíveis falhas no sistema, e a partir daí iniciar uma invasão. O primeiro alvo selecionado foi o chip da Figura 12 (b), o chip MX25L3206E. Sendo uma memória flash SPI (*Serial Peripheral Interface*), esse tipo de chip geralmente é utilizado para armazenar dados que necessitam ser mantidos mesmo quando o sistema encontra-se desligado, como por exemplo configurações da BIOS. A BIOS (Basic Input/Output System) é responsável por inicializar o hardware do computador e fornecer uma interface básica para configurações de sistema. O *firmware* da BIOS é carregado do chip para a memória do sistema durante a inicialização.

Sendo assim, pode-se observar se o MX25L3206E checa a própria integridade. Modificando a memória EEPROM do chip manualmente e o reinserindo na placa-mãe, pode-se analisar se tais mudanças na configuração da BIOS afetam o processo de inicialização do computador. Dessa forma, compromete-se toda a cadeia de autenticação do sistema PowerEdge R710.

A princípio, a ideia é simplesmente alterar *strings* que aparecem durante a inicialização para verificar comprometimento do sistema, apenas como prova de conceito. Foge do escopo do trabalho a realização de um ataque real e suas possíveis escaladas de privilégio, visto que o objetivo é o comprometimento da cadeia de produção em si.

2.2.1 Chip MX25L3206E

2.2.1.1 Escrita e Gravação na Memória EEPROM

Inicialmente, a ideia é dessoldar o chip MX25L3206E da placa-mãe 00NH4P durante o experimento. Em seguida, conectá-lo a um computador **Raspberry Pi 4 B** seguindo a pinagem adequada apontada pelo datasheet do chip em questão (MACRONIX, 2022), ilustrada na Figura 14.

Levando em conta a pinagem do Raspberry Pi, a ideia é fazer o circuito da Figura 15. Com esse circuito, pode-se utilizar o sistema operacional instalado no Raspberry para

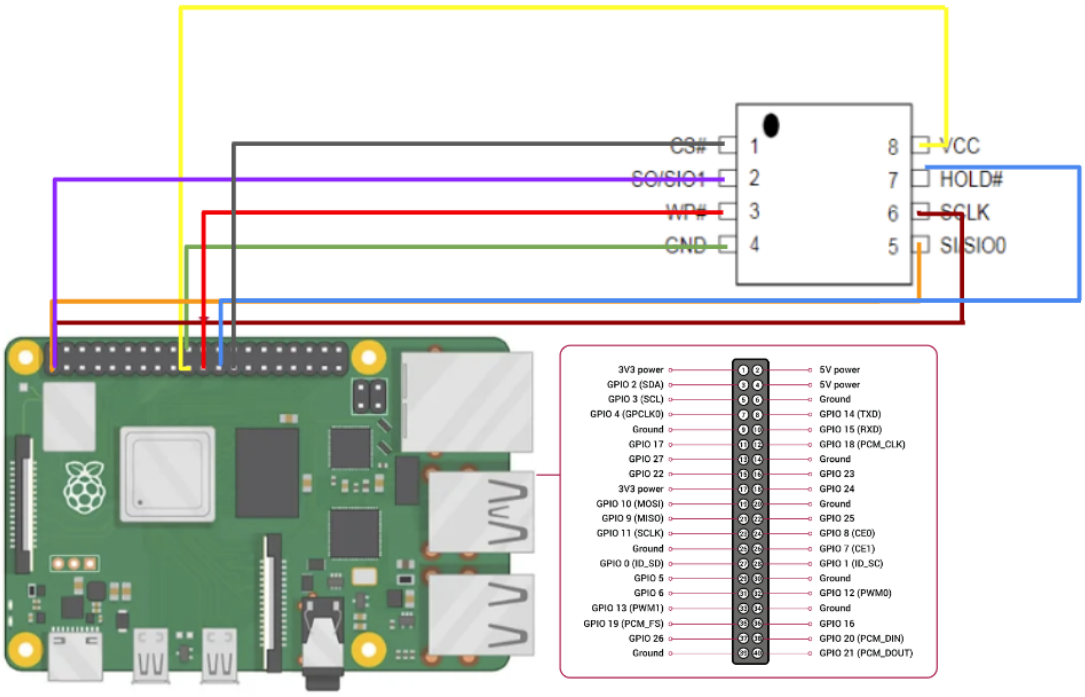


Figura 15 – Diagrama de conexão do Raspberry Pi com o chip.

ler e gravar na memória EEPROM do chip em questão.

Para fazer a gravação, utiliza-se do Flashrom, um software de código aberto projetado para ler e gravar memória flash em dispositivos eletrônicos (ROTOTRON, 2018). Ademais, como visto na documentação online do próprio software (FLASHROM, 2020), o Flashrom suporta o chip MX25L3206E com o qual o trabalho está sendo realizado.

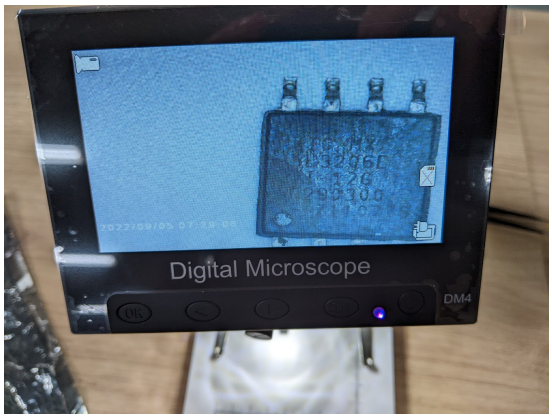
2.2.1.2 Prova de Conceito

Com a memória flash em mãos, a ideia seria realizar um ataque completo e verificar o nível de comprometimento do sistema. Porém, para o escopo desse trabalho, será necessário somente uma mudança de *strings* (como por exemplo, na inicialização da máquina) para simbolizar a possibilidade de um ataque real. Ou seja, uma prova de conceito.

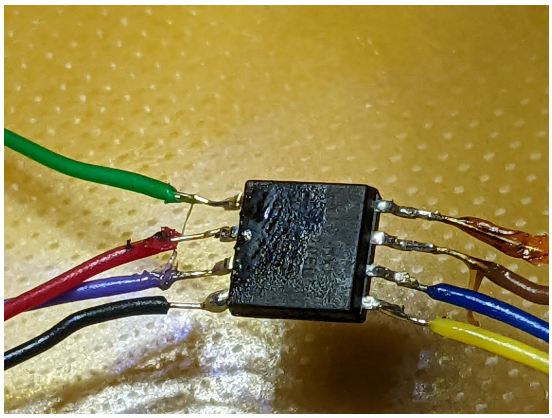
3 Modelo Experimental

3.1 Soldagem do chip MX25L3206E

Inicialmente, o chip MX25L3206E foi cuidadosamente dessoldado da placa-mãe 00NH4P. Como ele estava adesivado e por ter um tamanho pequeno, foi necessário iluminar adequadamente e ampliar digitalmente a imagem para identificar o tipo do chip do qual se tratava. O processo de limpeza e identificação do chip foi ressaltado na Figura 16 (a).



(a) Identificação do chip amplificada pelo microscópio digital.



(b) Zoom no chip após a soldagem realizada, já conectado ao Raspberry Pi.

Figura 16 – Identificação e soldagem do chip MX25L3206E

O chip, após cuidadosamente identificado e dessoldado, foi soldado a um Raspberry Pi 4 Modelo B, seguindo o diagrama da Figura 15. O chip pós-soldagem encontra-se na Figura 16 (b). Por fim, o resultado final encontra-se na Figura 17.

3.2 Modificação da memória do chip MX25L3206E

Com o chip dessoldado e conectado ao Raspberry Pi citado anteriormente, o procedimento seguinte foi utilizar o programa *Flashrom* para ler a memória flash do chip. Dentro do Sistema Operacional do Raspberry Pi, o comando de identificação foi utilizado:

```
1 eon@raspberrypi:~ $ flashrom -c "MX25L3206E/MX25L3208E" -p
    linux_spi:dev=/dev/spidev0.0,spispeed=20000 -V
2 flashrom v1.2 on Linux 6.1.21-v7+ (armv7l)
3 flashrom is free software, get the source code at https://
    flashrom.org
4
```

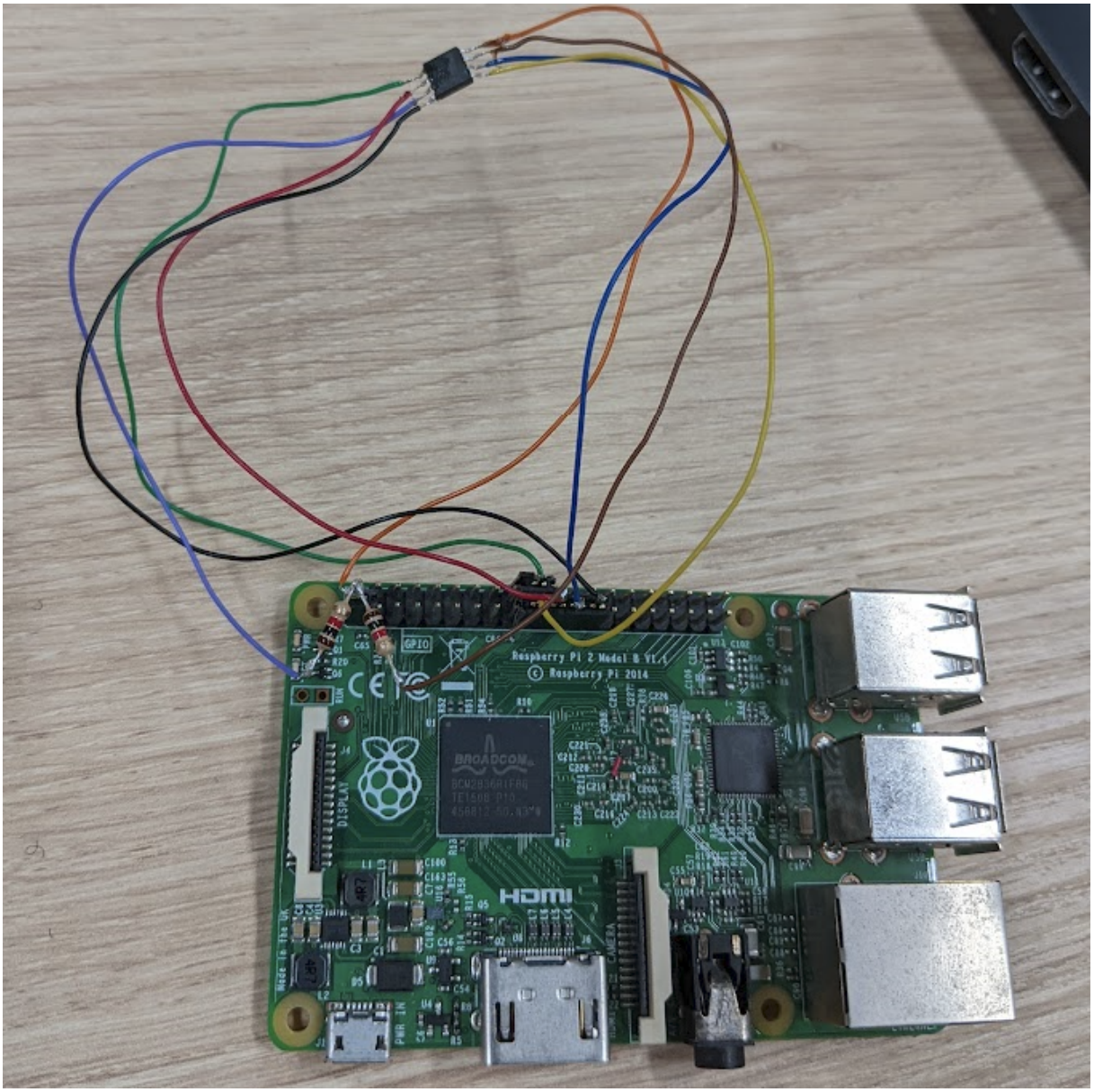


Figura 17 – Circuito do experimento feito de acordo com a Figura 15.

```
flashrom was built with libpci 3.6.4, GCC 9.2.1 20200123,
   little endian
Command line (5 args): flashrom -c MX25L3206E/MX25L3208E -p
   linux_spi:dev=/dev/spidev0.0,spispeed=20000 -V
Using clock_gettime for delay loops (clk_id: 1, resolution: 1
   ns).
Initializing linux_spi programmer
Using device /dev/spidev0.0
Using 20000kHz clock
linux_spi_init: Using value from /sys/module/spidev/
   parameters/bufsiz as max buffer size.
linux_spi_init: max_kernel_buf_size: 4096
```

```
13 The following protocols are supported: SPI.
14 Probing for Macronix MX25L3206E/MX25L3208E, 4096 kB:
     probe_spi_rdid_generic: id1 0xc2, id2 0x2016
15 Found Macronix flash chip "MX25L3206E/MX25L3208E" (4096 kB,
     SPI) on linux_spi.
16 Chip status register is 0x20.
17 Chip status register: Status Register Write Disable (SRWD,
     SRP, ...) is not set
18 Chip status register: Bit 6 is not set
19 Chip status register: Block Protect 3 (BP3) is set
20 Chip status register: Block Protect 2 (BP2) is not set
21 Chip status register: Block Protect 1 (BP1) is not set
22 Chip status register: Block Protect 0 (BP0) is not set
23 Chip status register: Write Enable Latch (WEL) is not set
24 Chip status register: Write In Progress (WIP/BUSY) is not set
25 This chip may contain one-time programmable memory. flashrom
     cannot read
26 and may never be able to write it, hence it may not be able
     to completely
27 clone the contents of this chip (see man page for details).
28 No operations were specified.
```

O comando acima basicamente executa o comando `flashrom` com algumas especificações de leitura.

A opção `-c "MX25L3206E/MX25L3208E"` indica ao programa procurar chips dos modelos MX25L3206E ou MX25L3208E.

A flag `-p linuxspi:dev=/dev/spidev0.0,spispeed=20000` indica que o programa está configurado para usar uma interface SPI no sistema Linux. O dispositivo SPI usado é /dev/spidev0.0, e a velocidade de comunicação SPI é definida como 20000 Hz (20 kHz).

Por fim, o `-V` indica que deseja-se visualizar a memória flash e seu conteúdo. Ao que tudo indica, o chip foi encontrado corretamente. Dessa forma executa-se um novo comando para salvar o conteúdo num arquivo .bin:

```
1 flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=2000 -r
    initial_flash_mem.bin
```

Entretanto, ao analisar a memória lida pela flashrom, o processo foi inconclusivo: o *hexdump* consistia somente de sequências de trechos `16 C2 20`, claramente indicando

corrupção na leitura. Após investigado, descobriu-se que mesmo que a versão do programa mais atual alegasse suportar o modelo do *chip* de acordo com a documentação técnica, o programa apresentava mensagem de erro a respeito de uma suposta proteção de blocos e não lia a memória corretamente. Dessa forma, fez-se necessário buscar uma alternativa. Para isso, utilizou-se um gravador CH341A Pro (EATON, 2023).

O CH341A Pro conecta-se ao computador por meio de uma porta USB e é capaz de ler, escrever e apagar dados em dispositivos de memória, permitindo a modificação de configurações, reparo de dispositivos danificados ou até mesmo a recuperação de placas-mãe com problemas de *firmware*. Ademais, utiliza-se o programa *CH341A Programmer*. Esse software oferece uma interface gráfica para leitura, gravação e apagamento de memórias EEPROM em dispositivos eletrônicos. Essa interface é apresentada na Figura 18.

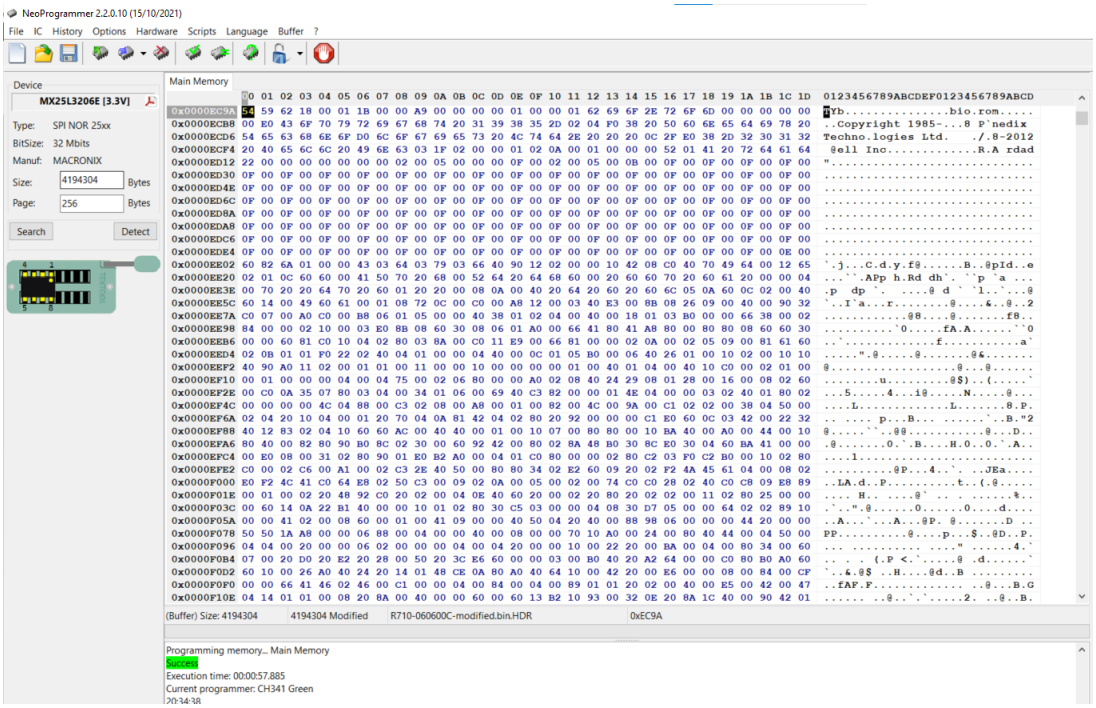


Figura 18 – Interface gráfica do Programador CH341A.

O *dump* é muito grande para ser mostrado por completo, mas um detalhe crucial dele foi entender que existiam algumas *strings* que haviam sido comprimidas, mas não cifradas. Pode-se observar isso devido ao formato das *strings*, que formam textos com alguns caracteres fora do lugar, como mostrado na Figura 19. Na imagem, pode-se observar a formação de algumas mensagens nesse formato, como “Press any key to. continue” e “Invalid m@emory”.

Um fator importante é notar que não é possível simplesmente alterar o arquivo comprimido e gravá-lo na memória flash. Como desconhece-se o algoritmo de compressão, pode ser que alterar os caracteres livremente gere erros e/ou comportamentos inesperados. Dessa forma, deve-se utilizar alguma ferramenta de descompressão desse formato. No caso,

```
0008A640:  61 32 E4 CD 16 02 3C 00   F8 72 8E F0 50 72 65 73  a2....<..r..Pres
0008A650:  73 20 61 6E 79 20 6B 65   79 20 74 6F B0 20 63 6F  s any key to. co
0008A660:  6E 74 69 6E 75 65 2E 2E   2E 02 68 F0 45 72 72 6F  ntinue....h.Erro
0008A670:  72 3A 20 49 6E 76 61 6C   69 64 20 6D 40 65 6D 6F  r: Invalid m@emo
0008A680:  72 79 03 23 F0 66 69 67   75 72 61 74 69 6F 6E 20  ry.#.figuration
0008A690:  64 65 74 65 63 01 02 20   64 2E 20 02 99 00 4D 05  detec.. d. ...M.
0008A6A0:  21 01 30 20 69 74 69 02   32 00 7A 05 22 80 57 61  !.0 iti.2.z."."Wa
0008A6B0:  72 6E 69 6E 67 3A 20 06   1E F0 73 69 7A 65 20 6D  rning: ...size m
0008A6C0:  61 79 20 62 65 20 72 65   64 75 00 63 02 3D 02 D6  ay be redu.c.=..
0008A6D0:  70 55 6E 73 75 70 70 6F   72 02 4C 0F 6A 04 6A 01  pUnsuppor.L.j.j.
0008A6E0:  61 F0 20 44 49 4D 4D 20   6D 69 73 6D 61 74 63 68  a. DIMM mismatch
0008A6F0:  20 61 20 63 72 6F 02 BE   02 78 40 73 6C 6F 74 73   a cro...x@slots
0008A700:  09 8A 12 23 08 73 0F 55   0F C0 0D C0 10 54 68 02  ...#.s.U.....Th.
0008A710:  99 0F E6 03 E6 01 73 B0   20 6E 6F 74 20 6F 70 74  ......s. not opt
0008A720:  69 6D 61 6C 03 EC 03 2A   11 3E 11 30 40 6D 6D 65  imal...*.>.0@mme
0008A730:  6E 64 0F B2 09 36 11 3E   12 B1 04 B7 1D 37 12 3A  nd...6.>.....7.:
0008A740:  10 65 61 02 C4 80 43 50   55 20 73 68 6F 75 6C 12  .ea...CPU shoul.
0008A750:  5F 03 D5 14 47 1B 0A 1C   6E 11 8C 30 00 20 20 20  _...G...n..0.
```

Figura 19 – Seção do *hexdump* do arquivo `initial_flash_mem.bin` onde pode-se observar algumas *strings* formando frases entendíveis.

arquivos desse conjunto de BIOS de servidores é abrangido pela *PhoenixTool* (ANDYP, 2022). A interface da ferramenta está ilustrada na Figura 20.

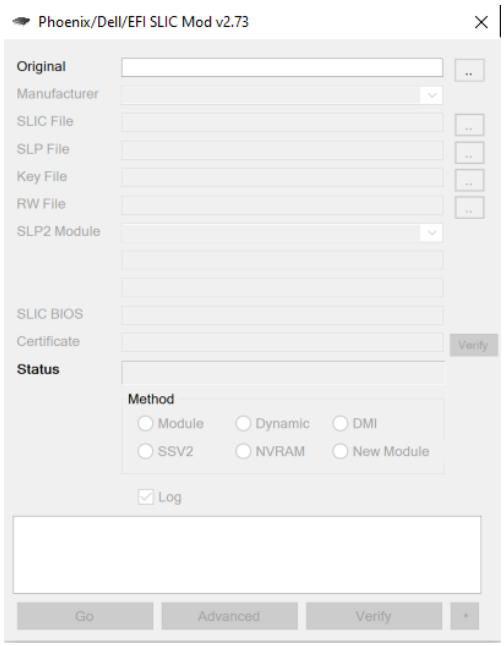


Figura 20 – Interface Gráfica da PhoenixTool.

Porém, ao tentar aplicar a ferramenta PhoenixTool para descomprimir o arquivo extraído da memória *flash*, observa-se que o formato do arquivo está incorreto e a operação não pode ser finalizada.

Investigando exemplos de uso da ferramenta PhoenixTool, percebeu-se que ela não foi criada para trabalhar com arquivos lidos e gravados diretamente na *flash* dessoldada da placa-mãe. Em vez disso, ela trabalha com arquivos em formato `.hdr`, que são empacotados dentro do software de atualização da BIOS fornecido oficialmente pela Dell em seu site.

Essa decisão dos desenvolvedores da PhoenixTool ocorreu, muito provavelmente, para tornar sua utilização mais simples para o usuário final sem experiência com soldagem, visto que o propósito original da ferramenta era auxiliar na pirataria de produtos da Microsoft, por meio da alteração de chaves de licença contidas na BIOS.

Comparando o arquivo extraído da *flash* com o arquivo `.hdr` contido no atualizador para a última versão da BIOS de servidores Dell PowerEdge R710 encontrado no próprio site da Dell, observa-se que a diferença reside na existência de um cabeçalho de 64 bytes no arquivo `.hdr` que não é gravado na *flash* e, portanto, serve apenas para fornecer informações ao software atualizador. Havia também dois trechos de algumas centenas de bytes que estavam preenchidos com informações somente no arquivo extraído da *flash*, mas estavam vazios no arquivo `.hdr`. Esses trechos aparentavam conter configurações realizadas no SETUP da máquina.

Inicialmente, tentou-se copiar o cabeçalho do arquivo `.hdr` desempacotado do instalador para o início do arquivo extraído da *flash*, porém passá-lo como entrada à PhoenixTool resultou num erro de *checksum* incorreto. Para não ser necessário estudar a forma de cálculo do *checksum*, adotou-se como entrada o arquivo `.hdr` desempacotado do instalador, que foi renomeado para `initial_mem_bin.hdr` e está ilustrado na Figura 21.

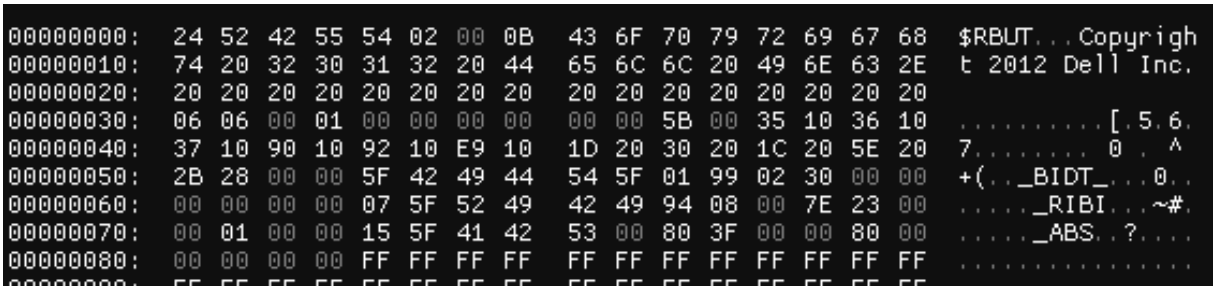


Figura 21 – Cabeçalho incluso no arquivo `initial_mem_bin.hdr`.

Com o arquivo no formato esperado, o processo de descompressão da ferramenta Phoenix Tool foi iniciado. Durante a execução da ferramenta, ela realiza diversas operações de adaptação do arquivo comprimido da memória. Entretanto, visando realizar o mínimo de mudanças possível, foram selecionadas somente as opções de descompressão e possibilidade de modificação dos arquivos antes do processo de *repacking* (recompressão). As opções selecionadas durante a execução da ferramenta estão na Figura 22.

Após início da execução da ferramenta, faz-se a descompressão dos vários módulos do arquivo numa pasta DUMP. O programa então libera o usuário para realizar mudanças antes da recompressão. Isso está sendo mostrado na Figura 23.

Após finalizado o processo, foi possível observar o arquivo em seu formato original dividido em módulos numa pasta inteira com diversos arquivos contendo trechos descomprimidos do código. As *strings*, antes previamente interrompidas por certos caracteres, agora podem ser lidas na íntegra. Um exemplo foi mostrado na Figura 24.

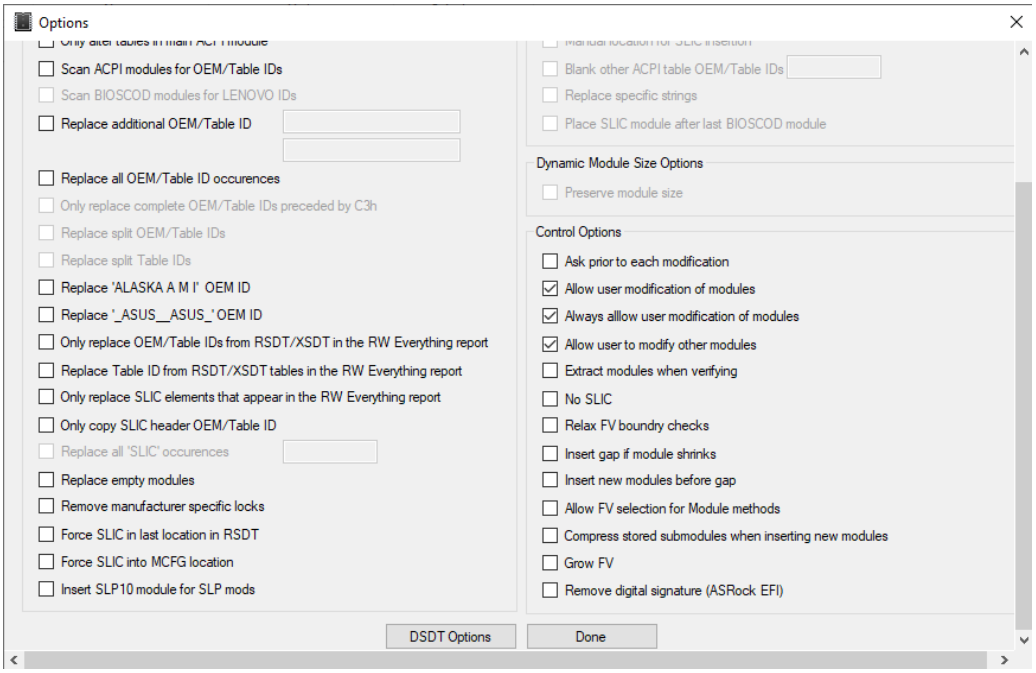


Figura 22 – Opções de execução da ferramenta PhoenixTool.

Com o programa tendo liberado as alterações manuais, pode-se fazer as duas alterações principais para a prova de conceito deste trabalho. Todas as *strings* “Dell”, como por exemplo a destacada na Figura 25, foram alteradas para *strings* “Hell”, como indicado na Figura 26.

Uma mudança manual similar à citada acima foi alterar todas as trings “Phoenix”, como por exemplo a da Figura 27, para “Pwnedix”, como mostrado na Figura 28. Um fator importante a ser notado é a conservação do tamanho da palavra, de forma que a mudança não altere o tamanho do arquivo nem os endereços referenciados pelo código. Isso é importante para que o chip seja regravado corretamente.

Com essas mudanças feitas, a PhoenixTool finaliza a recompressão e gera o arquivo `modified_flash_mem.bin`. Novamente, utiliza-se o gravador CH 314A Pro (EATON, 2023), semelhante ao processo de leitura, junto ao *CH341A Programmer* da Figura 18.

Utilizando o programa, bastou plugar o chip conectado via fios ao gravador CH341A Pro e realizar sua gravação. Além disso, o cabeçalho mencionado acima e ilustrado na Figura 21 foi removido para conservar o tamanho do arquivo `modified_flash_mem.bin`, mantendo-o idêntico ao de `initial_flash_mem.bin`. A gravação foi feita corretamente e o procedimento foi ilustrado na Figura 29.

Ademais, após a gravação ter sido finalizada, o chip MX25L3206E foi ressoldado no mesmo lugar na placa-mãe 00NH4P. A ideia é que a memória *flash* inicie e prossiga normalmente, com apenas as alterações simbólicas “Hell” e “Pwnedix” substituindo a maioria das *strings* apresentadas durante a inicialização do aparelho.

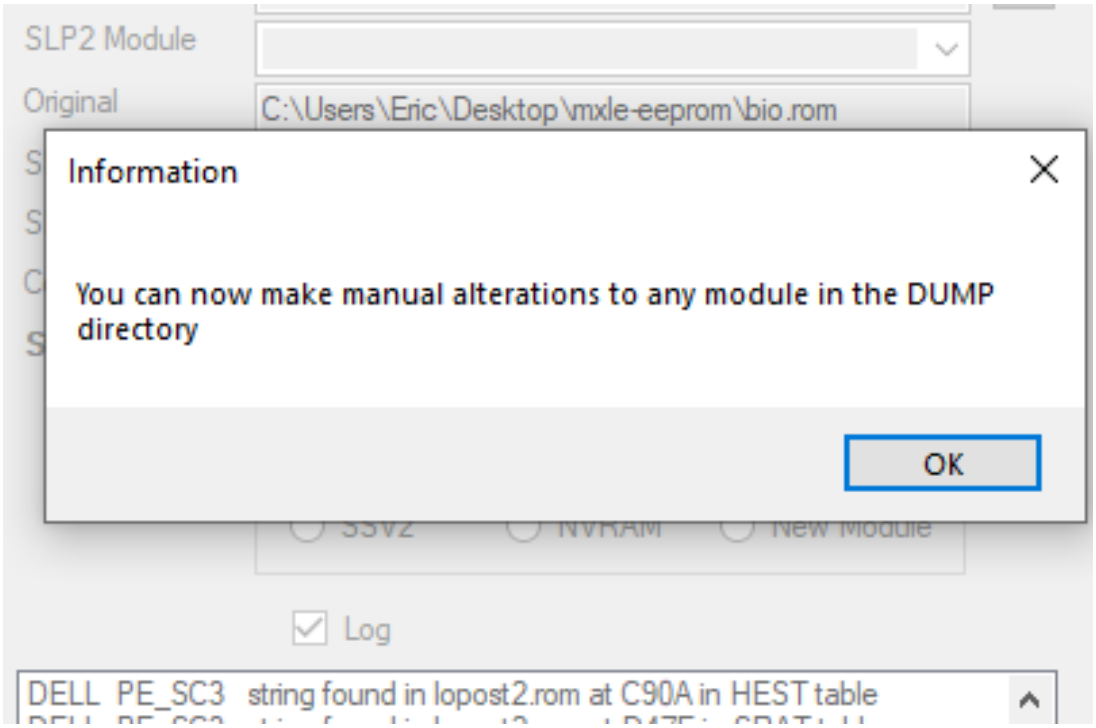


Figura 23 – Mensagem de liberação para alterações na memória decomprimida da memória *flash*. Sua tradução direta é "Você agora pode realizar alterações manuais para qualquer módulo no diretório DUMP".

```
00003AD0:  E8 9F 9F 0D 0A 4C 6F 63  61 6C 20 6B 65 79 62 6F  .....Local keybo
00003AE0:  61 72 64 20 6D 61 79 20  6E 6F 74 20 77 6F 72 6B  ard may not work
00003AF0:  20 62 65 63 61 75 73 65  20 61 6C 6C 20 75 73 65   because all use
00003B00:  72 20 61 63 63 65 73 73  69 62 6C 65 20 55 53 42  r accessible USB
00003B10:  20 70 6F 72 74 73 20 61  72 65 20 64 69 73 61 62   ports are disab
00003B20:  6C 65 64 2E 0D 0A 49 66  20 6F 70 65 72 61 74 69  led...If operati
00003B30:  6E 67 20 6C 6F 63 61 6C  6C 79 2C 20 70 6F 77 65  ng locally, powe
00003B40:  72 20 63 79 63 6C 65 20  74 68 65 20 73 79 73 74  r cycle the syst
00003B50:  65 6D 20 61 6E 64 20 65  6E 74 65 72 20 73 79 73  em and enter sys
00003B60:  74 65 6D 20 73 65 74 75  70 20 70 72 6F 67 72 61  tem setup progra
00003B70:  6D 0D 0A 74 6F 20 63 68  61 6E 67 65 20 73 65 74  m..to change set
00003B80:  74 69 6E 67 73 2E 0D 0A  00 BE D5 38 E8 F7 9E E8  tings.......8....
00003B90:  E0 9E 0D 0A 53 74 72 69  6B 65 20 46 31 20 74 6F  ....Strike F1 to
00003BA0:  20 72 65 74 72 79 20 62  6F 6F 74 2C 20 46 32 20   retry boot, F2
00003BB0:  66 6F 72 20 73 79 73 74  65 6D 20 73 65 74 75 70  for system setup
00003BC0:  2C 20 46 31 31 20 66 6F  72 20 42 49 4F 53 20 62  , F11 for BIOS b
00003BD0:  6F 6F 74 20 6D 61 6E 61  67 65 72 2E 0D 0A 00 E8  oot manager.....
```

Figura 24 – *Strings* encontradas no arquivo da pasta de módulos descomprimidos `bio.rom`.

3.3 Resultados

Com a placa-mãe 00NH4P novamente reconectada no gabinete e o computador tendo sido iniciado normalmente, pode-se verificar as imagens ressaltando a inicialização ocorrendo normalmente apesar dele ter sido comprometido de acordo com o escopo da prova de conceito desse trabalho.

Na Figura 30 (a) pode-se verificar a ocorrência das *string* "Hell" (Substituindo "Dell") e "Pwnedix" (Substituindo "Phoenix"), enquanto na subfigura (b) pode-se observar

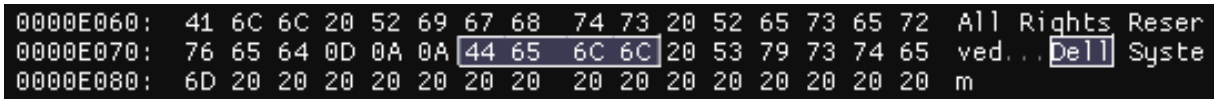


Figura 25 – *String* "Dell" encontrada no arquivo `rebrand.rom` antes da mudança.

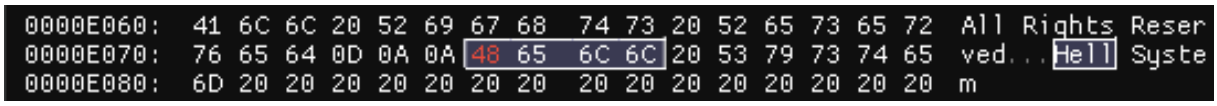


Figura 26 – *String* "Hell" encontrada no arquivo `rebrand.rom` depois da mudança.

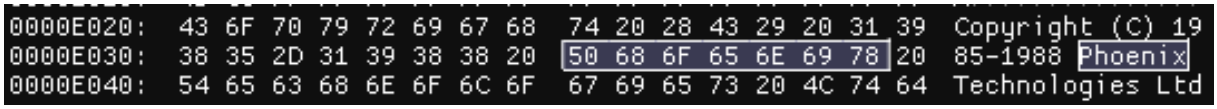


Figura 27 – *String* "Phoenix" encontrada no arquivo `rebrand.rom` antes da mudança.

como o processo de *boot* é concluído normalmente, dando espaço para uma inicialização normal do Sistema Operacional.

Foi necessário uma primeira reinicialização da máquina para que o boot ocorresse sem nenhuma mensagem de memória corrompida ou inicialização fora do normal. Porém, após esse *reboot* não ocorreu nenhuma mensagem (Fora as *strings* alteradas) que desse qualquer indício de alteração no *firmware*. Em geral, o experimento conseguiu atender às expectativas dentro do escopo definido durante a modelagem teórica.

É importante ressaltar que o módulo TPM (Trusted Platform Module, discutido na Seção 1) detecta alterações no código sendo executado no chip alterado, logo o ataque não passaria despercebido num cenário real. Estudar formas de fazer um ataque que passasse completamente em branco pode ser uma linha explorada futuramente, nos moldes de trabalhos de bypass do módulo como Han Wook Shin e Kim (2018), porém o conceito do ataque ainda é válido para máquinas que não possuam um módulo TPM.

Ademais, o resultado esperado também valida a identificação dos chips envolvidos no experimento, ressaltando a validade da investigação ocorrida na seção 2.1.2 sobre a placa-mãe do ambiente de testes.

```
0000E020:  43 6F 70 79 72 69 67 68  74 20 28 43 29 20 31 39  Copyright (C) 19
0000E030:  38 35 2D 31 39 38 38 20  50 77 6E 65 64 69 78 20  85-1988 Pwnedix
0000E040:  54 65 63 68 6E 6F 6C 6F  67 69 65 73 20 4C 74 64  Technologies Ltd
```

Figura 28 – *String* "Pwnedix" encontrada no arquivo `rebrand.rom` depois da mudança.

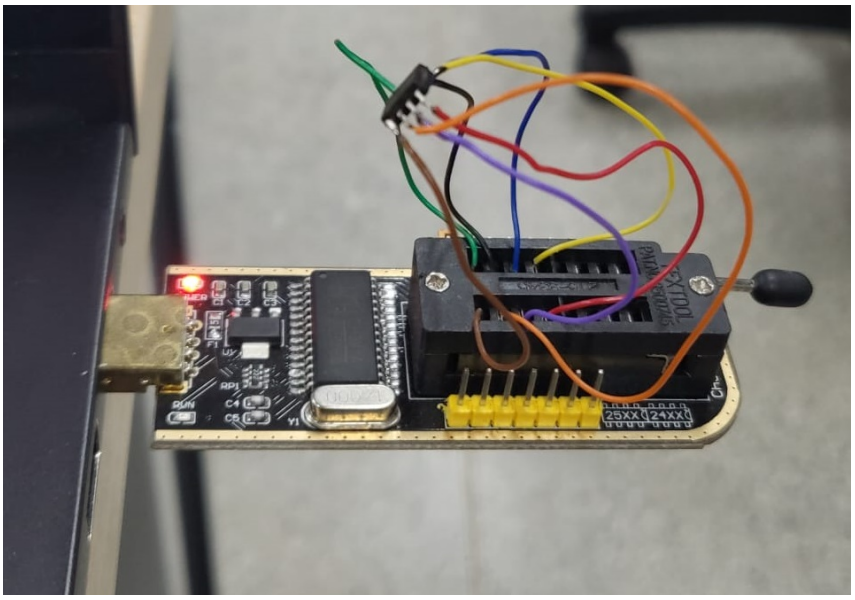
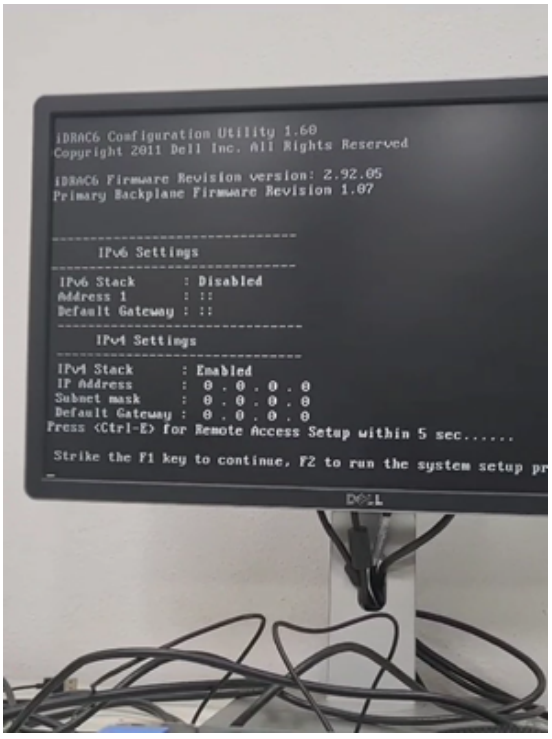


Figura 29 – Chip MX25L3206E conectado ao gravador CH314A Pro e plugado numa porta USB.

(a) *String* alteradas sendo mostradas com sucesso como parte da prova de conceito.



(b) Processo de boot sendo finalizado sem sinalização de mensagens de erro.

Figura 30 – Imagens do processo de inicialização do Dell PowerEdge R710 após ressoldagem do chip ao fim do experimento.

4 Conclusão

Esse trabalho apresentou uma prova de conceito de como infectar uma peça de *firmware* e como ela pode potencialmente comprometer uma máquina e a cadeia de produção que segue dessa unidade. Os passos são facilmente replicáveis com os mesmos softwares e ferramentas apresentadas no projeto, dadas as devidas especificações de cada placa-mãe e sistema como um todo.

A prova de conceito descrita no Capítulo 2 confirma o objetivo do Capítulo 1 de comprometer a cadeia e foi comprovada pelo experimento realizado no Capítulo 3. Observa-se uma coesão entre a parte teórica e a parte prática do projeto, ressaltando o procedimento feito.

É importante notar que o experimento foi realizado em um servidor real que, alguns meses antes do desenvolvimento deste trabalho, pertencia ao parque de máquinas da empresa parceira, demonstrando a aplicabilidade real das técnicas aqui exploradas.

4.1 Trabalhos Futuros

Mesmo que o projeto tenha sido concluído adequadamente para o escopo inicialmente descrito, existem melhorias para projetos futuros que podem advir dessa pesquisa. A primeira seria estudar quais ataques podem ser aplicados por esse procedimento e suas respectivas escaladas de privilégios, substituindo a prova de conceito apresentada no projeto.

Outra possível melhoria é estudar outros vetores de ataque. Durante o próprio Capítulo 2 são mencionados outros chips que seriam interessantes como porta de entrada. O chip escolhido foi o MXL253206E, porém chips BMC ou da própria placa de rede poderiam ter sido escolhidos e estudados para obter resultados semelhantes. Vários ataques conhecidos poderiam ter sido testados como os observados em (MATROSOV, 2019) e (FERN, 2021). Ademais, outros tipos de hardware também se encaixam nesse critério, ainda mais caso tenham a chance de realizar testes de tal natureza em placas e sistemas mais atuais e/ou mais complexos.

Referências

ANALOG DEVICES. *ADP3193A Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/183771/AD/ADP3193A.html>. Último acesso em: 27/08/2023. Citado na página 25.

ANDYP. *PhoenixTool v2.73*. 2022. <https://forums.mydigitallife.net/threads/tool-to-insert-replace-slic-in-phoenix-insyde-dell-efi-bioses.13194/>. Último acesso em: 25/08/2023. Citado na página 37.

ATMEL. *AT45DB041D Datasheet*. 2023. <https://dl.btc.pl/kamami_wa/at45db041d.pdf>. Último acesso em: 27/08/2023. Citado na página 27.

BROADCOM. *BCM5709 Datasheet*. 2023. <https://www.broadcom.com/products/ethernet-connectivity/network-adapters/bcm5720-1gbase-t-ic>. Último acesso em: 27/08/2023. Citado na página 27.

BYTEK. 2023. <https://www.amazon.com/DELL-00NH4P-POWEREDGE-System-Board/dp/B00HJKMO1M>. Último acesso em: 25/08/2023. Citado 2 vezes nas páginas 13 e 24.

DELL. 2022. <https://i.dell.com/sites/csdocuments/Business_solutions_engineering-Docs_Documents/en/poweredge-r710-technical-guidebook.pdf>. Último acesso em: 25/08/2023. Citado 2 vezes nas páginas 13 e 23.

DIODES INCORPORATED. *SRX7353 Datasheet*. 2023. <https://tz.key-components.com/parts/A07/SRX7353.html>. Último acesso em: 27/08/2023. Citado na página 27.

EATON. *CH341A Datasheet*. 2023. <https://datasheetspdf.com/datasheet/CH341A.html>. Último acesso em: 27/08/2023. Citado 2 vezes nas páginas 36 e 39.

FERN, N. Hardware: A Double-Edged sword for security. In: . [S.l.]: USENIX Association, 2021. Citado na página 43.

FLASHROM. *Supported hardware*. 2020. <https://wiki.flashrom.org/Supported_hardware>. Último acesso em: 25/08/2023. Citado na página 32.

HAN WOOK SHIN, J.-H. P. S.; KIM, H. A bad dream: Subverting trusted platform module while you are sleeping. *27th USENIX Security Symposium*, 2018. Citado na página 41.

IDT. *FGP202AKL Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/470078/IDT/9FGP202AKLF.html>. Último acesso em: 27/08/2023. Citado na página 26.

INFINEON. *020N03LS Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/507268/INFINEON/BSC020N03LSG.html>. Último acesso em: 27/08/2023. Citado na página 29.

JOTRIN. *9DB1200BGLF Datasheet*. 2023. <https://www.jotrin.com/product/parts/9DB1200BGLF>. Último acesso em: 27/08/2023. Citado na página 28.

JU FAN STORE. *100% novo std728 st d728 QFN-10 chipset*. 2023. <https://pt.aliexpress.com/item/1005004828241837.html>. Último acesso em: 27/08/2023. Citado na página 26.

MACRONIX. *MX25L3206E Technical Documentation*. 2022. <https://www.mxic.com.tw/en-us/support/design-support/Pages/technical-document.aspx>. Último acesso em: 25/08/2023. Citado na página 31.

MACRONIX. *MX25L3206E Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/575455/MCNIX/MX25L3206E.html>. Último acesso em: 27/08/2023. Citado na página 30.

MATROSOV, A. G. A. Breaking through another side: Bypassing firmware security boundaries from embedded controller. *Blackhat USA*, 2019. Citado na página 43.

MAXIM INTEGRATED. *8643AE Datasheet*. 2023. <https://pdf1.alldatasheet.net/datasheet-pdf/view/1372497/MAXIM/MAX38643AELT+.html>. Último acesso em: 27/08/2023. Citado na página 25.

NXP. *LM358A Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/17968/PHILIPS/LM358A.html>. Último acesso em: 27/08/2023. Citado na página 28.

ON SEMI. *NJT4030P Datasheet*. 2021. <https://www.onsemi.com/pdf/datasheet/njt4030p-d.pdf>. Último acesso em: 27/08/2023. Citado na página 29.

ON SEMI. *ADP4100 Datasheet*. 2023. <https://pdf1.alldatasheet.com/datasheet-pdf/view/351570/ONSEMI/ADP4100.html>. Último acesso em: 27/08/2023. Citado na página 25.

RAMAKRISHNA, S. *New Findings From Our Investigation of SUNBURST*. 2021. <https://orangematter.solarwinds.com/2021/01/11/new-findings-from-our-investigation-of-sunburst/>. Último acesso em: 07/08/2023. Citado 2 vezes nas páginas 17 e 20.

RENESAS. *41BC Datasheet*. 2023. <https://pdf1.alldatasheet.net/datasheet-pdf/view-marking/1048060/RENESAS/ISL6341BCRZ.html>. Último acesso em: 27/08/2023. Citado na página 30.

RENESAS. *K0389 Datasheet*. 2023. <https://pdf1.alldatasheet.net/datasheet-pdf/view-marking/313688/RENESAS/RJK0389DPA.html>. Último acesso em: 27/08/2023. Citado na página 29.

ROTOTRON. *Recover Bricked BIOS using FlashRom on a Raspberry Pi*. 2018. <https://www.rototron.info/recover-bricked-bios-using-flashrom-on-a-raspberry-pi/>. Último acesso em: 27/08/2023. Citado na página 32.

SHEN, C. et al. Research on trusted computing and its development. *Science China Information Sciences*, 2010. Citado 3 vezes nas páginas 13, 19 e 20.

SHOP233861. *K0204 KO204 KO2O4 5*. 2023. <https://pt.aliexpress.com/item/32334270137.html>. Último acesso em: 27/08/2023. Citado na página 26.

TEXAS INSTRUMENTS. *SN74CBTLV3253 Datasheet*. 2018. <https://www.ti.com/lit/ds/symlink/sn74cbtlv3253.pdf>. Último acesso em: 27/08/2023. Citado na página 29.

TEXAS INSTRUMENTS. *TMPx75 Datasheet*. 2020. <https://www.ti.com/lit/ds/symlink/tmp175.pdf>. Último acesso em: 27/08/2023. Citado na página 29.