Lista de ilustrações

Sumário

1 Introdução

A crescente integração de dispositivos inteligentes em nossas vidas cotidianas, exemplificada pelo Amazon Echo, trouxe inúmeras conveniências, mas também suscitou preocupações significativas quanto à segurança e privacidade dos usuários. A capacidade do Amazon Echo de interagir por meio de comandos de voz e de se conectar à Internet para acessar uma variedade de serviços levanta questões sobre como os dados do usuário são coletados, armazenados e protegidos. Nesse contexto, este trabalho se propõe a investigar e demonstrar as potenciais vulnerabilidades de segurança e privacidade do Amazon Echo, aplicando técnicas de Aprendizado de Máquina (ML) para análise do tráfego de rede gerado pelo dispositivo.

Ao explorar os dados gerados pelo Amazon Echo, pretendemos identificar padrões e anomalias que possam indicar falhas na segurança da privacidade. O uso de técnicas de ML, como classificação e agrupamento de dados, permitirá analisar grandes volumes de informações e descobrir insights ocultos que podem revelar potenciais brechas na proteção de dados sensíveis dos usuários. Ao demonstrar essas falhas de segurança, espera-se destacar a importância de medidas robustas de proteção da privacidade e incentivar o desenvolvimento de soluções mais seguras para dispositivos IoT como o Amazon Echo.

2 Coleta de Dados

O projeto envolveu a coleta e análise de dados provenientes de interações de voz com a Amazon Alexa. Para isso, foram executados e registrados traços de 100 comandos de voz diferentes, realizados em duas línguas distintas, inglês e espanhol. Cada comando foi executado com três vozes sintéticas diferentes, disponíveis no serviço Amazon Polly. Além disso, foram coletadas 500 amostras diferentes de cada combinação de comando de voz e voz sintética, totalizando 300.000 traços de rede diferentes [BARCELO-ARMADA; CASTELL-UROZ; BARLET-ROS, 2022].

Os dados foram organizados em dois arquivos, um para cada idioma utilizado na coleta das medidas. Cada arquivo segue uma estrutura de pastas com dois níveis, onde o primeiro nível denota a voz sintética usada para executar o comando, e o segundo nível indica o comando realizado. Dentro de cada pasta de comando estão contidos arquivos PCAP brutos, que incluem toda a comunicação de rede entre o dispositivo Amazon Echo Dot e os servidores da Amazon. Cada arquivo PCAP consiste em uma lista de todos os pacotes de rede que passam pela interface de rede durante a captura ativa, incluindo tanto os cabeçalhos TCP quanto o payload do pacote. Também são registradas informações sobre o tempo de cada comunicação. É importante observar que muitas comunicações são codificadas usando TLS (Transport Layer Security), limitando a quantidade de informações do payload acessíveis pelo usuário.

Quando um usuário se comunica com a assistente virtual Alexa através de um alto-falante inteligente ou um smartphone, eles interagem com diferentes componentes do sistema Amazon Alexa. O primeiro componente é o próprio alto-falante inteligente, que reage às vozes dos comandos executados pelo usuário. O alto-falante inteligente tem poder computacional limitado e depende dos servidores do Alexa Voice Service (AVS) para decifrar a linguagem natural e interpretar o comando dado. Uma vez que o AVS sabe a ação a ser executada, ele chama a habilidade necessária (aplicações Alexa) através dos servidores AWS Lambda, especializados para a tarefa. Quando a habilidade ou tarefa é concluída, os servidores Lambda notificam o servidor AVS e este responde ao alto-falante inteligente, enviando a resposta do comando no formato de texto. Por fim, o alto-falante inteligente usa um sistema de texto para fala para ler a resposta em formato de áudio. Todos esses componentes compõem o ecossistema da Amazon Alexa e garantem uma resposta adequada aos comandos de voz emitidos pelo usuário.

O principal objetivo do projeto é coletar as comunicações entre o alto-falante inteligente e o servidor AVS. Para obter todos os traços de rede, foi realizado um tipo de ataque ”homem-no-meio”, forçando todas as comunicações a passarem por um ponto de

acesso controlado. No experimento de laboratório, o ponto de acesso consistia em um Raspberry Pi configurado como um ponto de acesso Wi-Fi. O dispositivo Amazon Echo Dot estava conectado ao Wi-Fi gerado pelo Raspberry Pi, e as comunicações chegavam aos servidores AVS através da porta LAN do Raspberry Pi. A Figura 1 mostra o processo de coleta de dados.

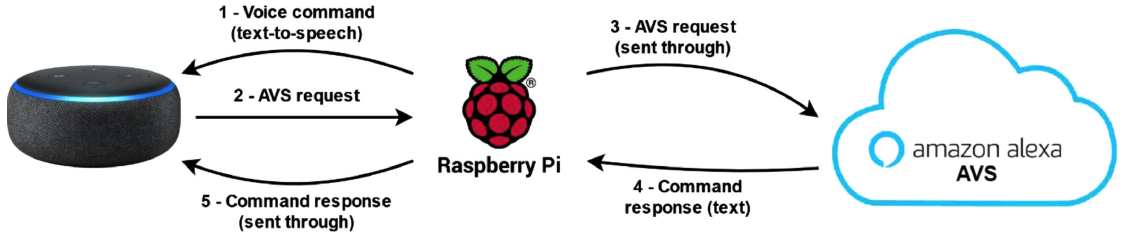

Figura 1 – Método para Coleta de Dados

Fonte: Barcelo-Armada, Castell-Uroz e Barlet-Ros [2022]

O processo consiste em seis etapas:

- O Raspberry Pi inicia o processo de captura de traços através da ferramenta TCP-Dump e gera um comando de voz. Para isso, ele seleciona um comando de voz por vez de um arquivo contendo uma lista dos comandos selecionados. O comando é reproduzido em voz alta usando o serviço de texto para fala em nuvem Amazon Polly, através dos alto-falantes conectados ao Raspberry Pi.
- O Amazon Echo Dot desperta, captura o comando de voz e o envia para os serviços AVS.
- O TCPDump em execução dentro do Raspberry Pi intercepta a solicitação.
- O Amazon AVS envia de volta a resposta que, mais uma vez, é interceptada pelo TCPDump.
- A resposta chega ao dispositivo Echo, onde é convertida em uma mensagem de áudio pelo próprio sistema de texto para fala.
- Antes de lançar o próximo comando de voz, o Raspberry Pi interrompe a captura do TCPDump e a salva como um arquivo PCAP de amostra dentro da pasta correspondente.

Para evitar ruídos e influências externas nas medidas, o único dispositivo conectado ao ponto de acesso Wi-Fi foram os alto-falantes inteligentes. Além disso, atualizações automáticas do Raspberry Pi OS e serviços desnecessários foram desativados para eliminar tráfego de rede não esperado. Além disso, foi necessário isolar o dispositivo o máximo possível de sons e vozes externas que pudessem fazê-lo despertar em intervalos inesperados. Para isso, o dispositivo foi configurado em um ambiente completamente fechado.

É importante ressaltar que apenas 500 amostras de cada uma das 13 classes foram escolhidas para a análise, resultando em um total de 6500 arquivos PCAP, com um tamanho total de 782 MB. Essa seleção foi realizada para reduzir o tempo necessário para o treinamento do modelo.

A Figura 2 e a Figura 3 mostram, respectivamente, exemplos de tipos de comandos e de arquivos PCAP capturados.

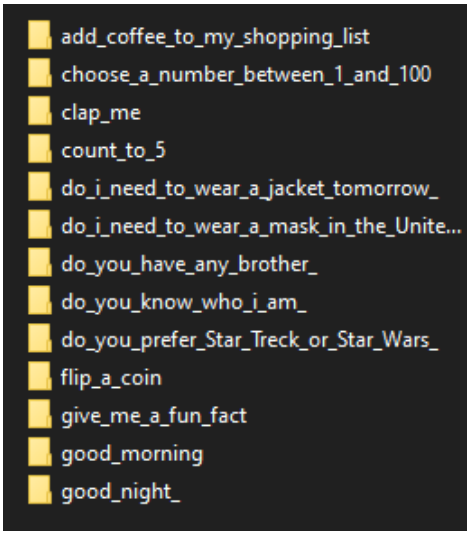

Figura 2 – Tipos de Comandos

Fonte: o autor

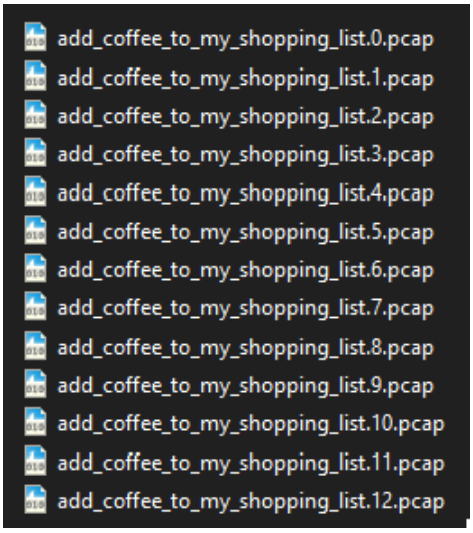

Figura 3 – Exemplos de arquivos PCAP coletados

Fonte: o autor

3 Extração de Features

Utilizando a biblioteca Scapy, do Python, foram extraídas 17 features de cada um dos arquivos PCAPs:

- Total packets sent: Esta característica simplesmente conta o número total de pacotes enviados durante a comunicação. É uma medida fundamental que fornece uma visão geral da atividade da rede.
- Pushed data packets: Refere-se ao número de pacotes que contêm dados que foram transmitidos durante a comunicação. Esses pacotes são aqueles que carregam informações úteis, como solicitações HTTP, respostas, etc.
- ACK packets sent: Indica o número de pacotes ACK (Acknowledgment) enviados. Os pacotes ACK são usados para confirmar a recepção de dados e manter o controle de fluxo.
- Pure ACKs sent: São os pacotes que contêm apenas informações de ACK, ou seja, não transportam dados reais. Eles são usados exclusivamente para confirmação e controle de fluxo.
- Actual data packets: Representa o número de pacotes que contêm dados reais, excluindo os pacotes que são apenas ACKs.
- Actual data bytes: Esta característica conta o número total de bytes de dados transmitidos, fornecendo uma medida quantitativa do volume de dados transmitidos durante a comunicação.
- Average window advance: Refere-se ao avanço médio da janela de transmissão durante a comunicação. A janela de transmissão é um mecanismo usado para controlar o fluxo de dados entre o emissor e o receptor.
- Maximum segment size: Indica o tamanho máximo dos segmentos de dados transmitidos durante a comunicação. Isso é importante para entender as limitações e capacidades da rede em termos de tamanho de pacote.
- Minimum segment size: Ao contrário da característica anterior, esta indica o tamanho mínimo dos segmentos de dados transmitidos.
- Initial window bytes: Refere-se ao número de bytes enviados inicialmente ao iniciar uma comunicação. Isso pode afetar o desempenho inicial da transmissão de dados.

- Data transmit time: Representa o tempo total de transmissão dos dados, proporcionando insights sobre a eficiência e velocidade da comunicação.
- Maximum idle time: Indica o período máximo de inatividade durante a comunicação, ou seja, o tempo máximo entre transmissões de dados.
- Maximum window advance: Similar à característica Average window advance, esta indica o avanço máximo da janela de transmissão durante a comunicação.
- Minimum window advance: Ao contrário da característica anterior, esta indica o avanço mínimo da janela de transmissão.
- Unique bytes sent: Refere-se ao número de bytes únicos enviados durante a comunicação, ou seja, excluindo duplicatas. Isso pode ser útil para entender a diversidade e variabilidade dos dados transmitidos.
- Average segment size: Indica o tamanho médio dos segmentos de dados transmitidos durante a comunicação, fornecendo uma medida média do volume de dados por pacote.
- Throughput: É uma medida da taxa de transferência de dados, representando a quantidade de dados transmitidos por unidade de tempo. É uma característica fundamental para avaliar o desempenho da rede.

A partir da extração das características de uma amostra, é possível detectar os padrões que cada uma das classes têm em comum [LI; MOORE, 2007].

4 Avaliação do Modelo Gerado

Na avaliação do modelo, foi utilizado o algoritmo Random Forest para classificação, por ser o algoritmo mais adequado para este problema em específico [LI; MOORE, 2007]. Esta abordagem resultou em uma acurácia notável de 98%, indicando uma performance robusta na tarefa de classificação. Durante o processo de extração de características dos arquivos PCAP, houveram desafios relacionados à demora. Para superar essa questão, foi utilizada a biblioteca Joblib para salvar o modelo treinado, permitindo que a reutilização do modelo sem a necessidade de repetir o treinamento a cada execução.

A figura 4 mostra as principais métricas estatísticas para avaliação do modelo, enquanto a figura 5 mostra a matriz de confusão gerada para todas as classes.

Classification Report:

	precision	recall	f1-score	support
add_coffee_to_my_shopping_list	0.99	0.99	0.99	150
choose_a_number_between_1_and_100	0.99	0.98	0.99	150
clap_me	1.00	1.00	1.00	150
count_to_5	0.96	0.97	0.97	150
do_i_need_to_wear_a_jacket_tomorrow_	0.97	0.99	0.98	150
do_i_need_to_wear_a_mask_in_the_United_States_	1.00	0.97	0.99	150
do_you_have_any_brother_	0.96	0.97	0.97	150
do_you_know_who_i_am_	1.00	0.99	0.99	150
do_you_prefer_Star_Treck_or_Star_Wars_	0.98	0.99	0.99	150
flip_a_coin	1.00	0.97	0.99	150
give_me_a_fun_fact	0.95	0.99	0.97	150
good_morning	1.00	0.99	0.99	150
good_night_	0.99	0.99	0.99	150
accuracy			0.98	1950
macro avg	0.98	0.98	0.98	1950
weighted avg	0.98	0.98	0.98	1950

Figura 4 – Métricas de Avaliação

Fonte: o autor

Confusion Matrix:

```
[[148   1   0   1   0   0   0   0   0   0   0   0   0]
 [  1 147   0   0   0   0   0   0   2   0   0   0   0]
 [  0   0 150   0   0   0   0   0   0   0   0   0   0]
 [  1   0   0 146   0   0   1   0   1   0   1   0   0]
 [  0   0   0   0 148   0   0   0   0   0   2   0   0]
 [  0   0   0   0   1 146   0   0   0   0   3   0   0]
 [  0   0   0   4   0   0 146   0   0   0   0   0   0]
 [  0   0   0   0   2   0   0 148   0   0   0   0   0]
 [  0   0   0   0   1   0   0   0 149   0   0   0   0]
 [  0   0   0   1   0   0   2   0   0 146   0   0   1]
 [  0   0   0   0   1   0   1   0   0   0 148   0   0]
 [  0   0   0   0   0   0   0   0   0   0   2 148   0]
 [  0   0   0   0   0   0   2   0   0   0   0   0 148]]
```

Figura 5 – Matriz de Confusão

Fonte: o autor

5 Aplicação do Modelo para Classificar Novas Amostras Coletadas

Para fins práticos, o modelo treinado foi utilizado para classificar novas amostras de tráfego que não pertencem ao dataset previamente obtido, de modo a verificar seu desempenho em cenários reais. Para isso, foi utilizado um notebook para capturar o tráfego de rede gerado pela Alexa através do Wireshark.

O tráfego gerado tem o nome de captured.pcap e se refere ao tráfego gerado por um comando “good morning”. O modelo foi aplicado ao arquivo utilizando o comando “python classify_pcap.py captured.pcap”e obteve êxito na classificação.

6 Conclusão

Neste estudo, foi realizada a aplicação do algoritmo Random Forest na classificação de dados provenientes de interações de voz com assistentes virtuais, utilizando arquivos PCAP como fonte de informação. A obtenção de uma acurácia de 98% demonstra a eficácia e robustez do modelo desenvolvido, evidenciando sua capacidade de generalização e precisão na identificação de padrões nas comunicações de rede. Sendo assim, é possível concluir que, apesar de ser primordial para a proteção da privacidade de seus usuários, a criptografia aplicada nos dispositivos IoT não é suficiente para proteger todas as informações sensíveis.

Referências

BARCELO-ARMADA, R.; CASTELL-UROZ, I.; BARLET-ROS, P. Amazon alexa traffic traces. *Universitat Politecnica de Catalunya*, 2022. Citado 2 vezes nas páginas 2 e 3.

LI, W.; MOORE, A. W. A machine learning approach for efficient traffic classification. *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2007. Citado 2 vezes nas páginas 6 e 7.