Utilizando características de GPU para identificação de *Smartphones*

Rodrigo Morbach[1], Renato de Jesus Manzoni[1], Eduardo Gielamo Oliveira[1], Edwin Delgado[1]

[1]Instituto de Pesquisas Eldorado
Av. Alan Turing, 275, Cidade Universitária – Campinas – SP – Brasil

{rodrigo.morbach,eduardo.oliveira}@eldorado.org.br,rmanzoni@gmail.com,
edelgado@alumni.usp.br

Abstract. *The collection of hardware and software attributes from a computer with the sole purpose of identification is known as fingerprinting. Apart from device tracking, fingerprinting can also be used as a method for security enhancement and fraud prevention. This paper aims to explore hardware accelerated image rendering in mobile devices in order to use it as a fingerprinting. The results show that our method can be used as input for mobile device class identification.*

Resumo. *O processo de coleta de atributos de um computador com o propósito de identificação é conhecido como fingerprinting. Ainda que o fingerprinting possa ser utilizado com a finalidade de rastreamento do dispositivo, ele pode também ser empregado em métodos de reforço de segurança e prevenção de fraudes. Este trabalho explora a renderização de imagens aceleradas por hardware em aplicações móveis para fingerprinting de dispositivo. Os resultados mostram que o método proposto pode ser usado como insumo para identificação de classes de dispositivos móveis.*

1. Introdução

Criado inicialmente no contexto da *Web* [Eckersley 2010], a técnica de *fingerprinting* consiste na coleta de atributos de *hardware* e *software* por um agente cliente, no caso, o próprio navegador, e envio destes para um servidor com o propósito de geração de um *hash* que identifica o dispositivo. Dentre as informações coletadas estão: idioma, sistema operacional, *plugins* instalados, *cookies*, *Timezone*, fontes instaladas, *User-agent* e resolução de tela. O trabalho de [Eckersley 2010] mostrou que a combinação desses atributos é, na maioria das vezes, única.

A identificação do dispositivo permite combater ataques e fraudes na Internet, por exemplo, cadastros realizados por robôs, ataques de força bruta na autenticação, transações financeiras e *spammers*. Uma vez que o atacante é identificado, pode-se aplicar mecanismos que bloqueiam o dispositivo para uso do serviço. Mecanismos similares de bloqueio como IP também são eficazes no combate destes ataques, contudo, existe a possibilidade de ferir algum usuário legítimo se o IP é compartilhado. Por isso, o uso de técnicas de bloqueios de IP e *fingerprinting* podem ser combinadas para minimizar o impacto para o usuário legítimo.

A evolução de tecnologias *Web* permite fornecer conteúdo rico para o usuário, a saber: áudio, vídeo, e animações, viabilizadas por uma maior aproximação dos navegadores *Web* com os recursos de *hardware*. Explorando essa evolução, [Mowery and Shacham 2012] mostram que a renderização *offscreen* de textos e imagens utilizando APIs como *Canvas* e *WebGL*[1] resultam em um *fingerprinting* consistente e de alta entropia. Outros resultados acerca dessa hipótese são apresentados no trabalho de [Laperdrix, Rudametkin and Baudry 2016].

No contexto de aplicações móveis, identificadores explícitos como IMEI (*International Mobile Equipment Identifier*) e IMSI (*International Mobile Subscriber Identity*) sempre foram usados com o intuito de identificar dispositivos unicamente. No entanto, por questões de privacidade, os principais sistemas operacionais do segmento aplicaram restrições de acesso a tais informações. No iOS, o acesso ao IMEI e IMSI não são permitidos e no *Android* são protegidos por permissão de usuário.

Neste cenário, diferentes trabalhos [Ferreira, Christof and Jonke 2018, Wu et al. 2016] têm abordado o uso de *fingerprinting* em aplicações móveis considerando a coleta de atributos específicos do dispositivo, como modelo e fabricante, resolução de tela e capacidades de armazenamento e processamento. Estes atributos contribuem no *fingerprinting* do dispositivo, haja vista estes dados são estáticos e não sofrem mudança com o tempo.

Este trabalho apresenta os resultados de estudo preliminar sobre *fingerprintings* obtidos a partir de renderização de imagens *offscreen* em *smartphones* e *tablets Android* e iOS. No trabalho [Cheng and Wang. 2011] foi explorado o uso do GPU em dispositivos móveis para propósitos gerais, este trabalho foi o ponto de partida para explorar o uso das técnicas do GPU no contexto de aplicações móveis.

2. Trabalhos relacionados

Embora *smartphones* possuem diversas características que os tornam passíveis de identificação, como sensores (*e.g,* acelerômetro, giroscópio), antenas (*WiFi*, *Bluetooth*, *NFC*), além de características de *software*, trabalhos de pesquisa sobre *fingerprinting* neste cenário são relativamente recentes. Contudo, ainda que o *fingerprinting* não seja a finalidade, o trabalho de [Ferreira, Christof and Jonke 2018] mostrou que diversas bibliotecas comerciais utilizadas por aplicações *Android* coletam informações que podem ser utilizadas para identificar unicamente *smartphones* e *tablets*.

O trabalho de [Wu et al. 2016] descreve a análise de 38 identificadores implícitos disponíveis em *smartphones Android* que não requerem permissão do usuário para sua obtenção. Os autores aplicaram um algoritmo de seleção de características para definir os melhores atributos e conduziram um experimento com 2239 dispositivos, alcançando uma taxa de acerto de 99%. A renderização de imagens *offscreen* não fez parte do estudo dos autores.

Realizado exclusivamente no sistema iOS, o trabalho de [Kurtz et al. 2016] estudou a geração de *fingerprinting* com base em 29 configurações personalizadas do usuário, como lista de contatos, álbuns de fotos, aplicativos instalados e listas de

[1] https://developer.mozilla.org/docs/Web/API/WebGL_API

músicas. Mais do que identificar o dispositivo, o trabalho apresentou um método para identificação do usuário.

Conforme constatado por revisão e corroborado pelo trabalho de [Ferreira, Christof and Jonke 2018], o uso de *fingerprinting* por meio de renderização *offscreen* utilizando GPU não havia sido explorado em dispositivos móveis até o momento.

3. Preparação das imagens

As características das imagens utilizadas no experimento foram baseadas no trabalho de [Cao, Li and Wijmans 2017], que utilizou diferentes técnicas de computação gráfica para gerar imagens via navegador *Web* utilizando *WebGL*, e avaliar a unicidade do *fingerprinting* gerado. Utilizamos características das imagens com os melhores resultados encontrados pelos autores para definir um conjunto de imagens a serem renderizadas pelos *smartphones*. A teoria é que diferentes algoritmos de interpolação empregados nas diferentes etapas de renderização variam de uma unidade de processamento gráfico (GPU) para outra, aumentando a entropia da imagem resultante.

As imagens geradas são exibidas na Figura 1. Cada imagem é composta por dois modelos, um bidimensional e outro tridimensional, e possui uma característica que a distingue das demais, a saber:

a) não foi aplicada a técnica de *anti-aliasing* para suavizar as bordas dos modelos (ao aplicar zoom na imagem é possível observar contornos bem definidos). Diferentes cores são desenhadas na superfície do triângulo, aumentando a interpolação entre o *vertex shader* e o *fragment shader*.
b) aplicação de *anti-aliasing* para suavização de contornos.
c) emprego de *anti-aliasing* e aplicação de textura ao modelo do cubo.
d) alteração na luz ambiente e emprego de *anti-aliasing*.
e) modelos com *anti-aliasing* sobrepostos e cubo rotacionado.
f) modelos com *anti-aliasing* sobrepostos, cubo rotacionado e iluminação difusa. A luz difusa causa reflexo quando ilumina um objeto, resultando em sombra de um objeto sobre outro.

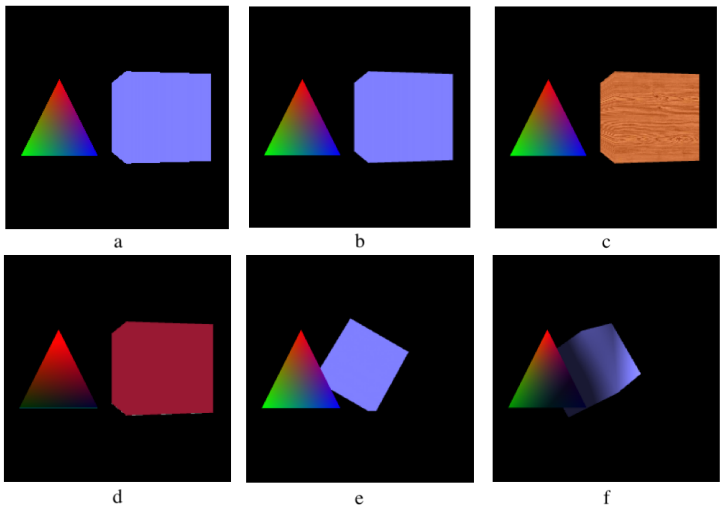

Figura 1. Imagens utilizadas no experimento.

4. Experimento

As plataformas *Android* e iOS possuem suporte nativo ao *OpenGL ES*[2], e fornecem APIs que permitem a renderização de imagens aceleradas por GPU. Além disso, ambas as plataformas permitem que a geração e a renderização da imagem aconteça de modo *offscreen*, isto é, sem que o usuário perceba que ela está de fato ocorrendo.

As etapas realizadas no cenário de experimentação são exibidas na Figura 2.

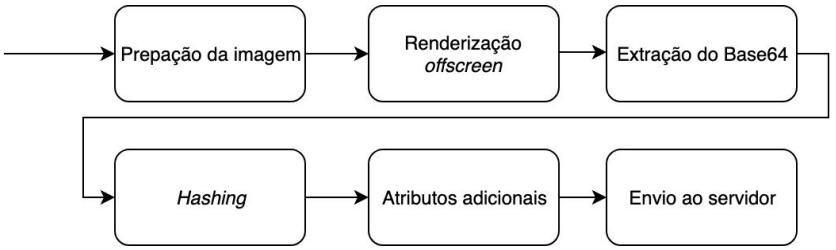

Figura 2. Etapas do processo de obtenção de fingerprinting.

As duas primeiras etapas compreendem o *pipeline* de renderização do *OpenGL* e ocorrem de modo *offscreen*. Após a renderização, a codificação *Base64* da imagem gerada é extraída e é realizado um *hashing* utilizando uma função MD5 para facilitar a comparação dos dados, além de diminuir a quantidade de informações trafegadas ao servidor. A etapa de adição de atributos compreende a inclusão de informações do dispositivo, tais como: modelo, fabricante e versão do sistema operacional.

Uma vez que a fragmentação de dispositivos é menor em aparelhos da *Apple*, decidiu-se implementar a renderização das imagens primeiramente no iOS. O processo descrito na Figura 2 foi realizado para as seis imagens apresentadas na Figura 1, sendo executadas em 11 dispositivos diferentes, compreendendo boa parte das unidades de processamento gráfico disponibilizadas pelo fabricante[3]. Após a coleta dos resultados do iOS, optou-se por implementar em *Android* a renderização da imagem que forneceu a maior entropia. Uma aplicação de teste foi disponibilizada para um grupo de usuários voluntários, dos quais os resultados são apresentados na próxima seção. O código fonte do experimento está disponível no *GitHub*[4].

5. Resultados

O método escolhido para avaliar a efetividade do *fingerprinting* foi o mesmo utilizado por [Mowery and Shacham 2012], [Wu et al. 2016] e [Eckersley, P. (2010)], a medida de entropia da distribuição dada pela Equação 1, onde $p(x_i)$ é a probabilidade do *fingerprinting* no universo de amostras coletadas.

$$E = -\sum_{i=1}^{n} p(x_i)\, log_2\, p(x_i) \quad (1)$$

[2] https://www.khronos.org/opengles/

[3] https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/HardwareGPUInformation/HardwareGPUInformation.html

[4] https://github.com/eld-rmorbach/GPUFingerprinting

A medida da unicidade é dada pela Equação 2, e indica quão único é o *fingerprinting* no universo de amostras coletadas.

$$u(x_i) = 1/p(x_i) \quad (2)$$

Conforme mencionado, inicialmente a renderização das imagens exibidas na Figura 1 foi realizada somente no iOS. Dentre os resultados gerados, a imagem **(e)** apresentou maior entropia, de 2.23 *bits*, enquanto que as demais apresentaram sempre o mesmo resultado, de 1.78 *bits*. Apesar de a amostragem considerar boa parte dos modelos de GPU disponíveis em aparelhos iOS, vale ressaltar que os valores obtidos não representam a real entropia dos testes, uma vez que a quantidade de dados é pequena se comparada ao universo global.

Em posse da imagem resultante em maior entropia na plataforma iOS **(e)**, sua renderização foi implementada em *Android*. Uma aplicação simples foi disponibilizada para usuários voluntários por meio de plataforma de testes *beta*, resultando na coleta de um total de 54 dispositivos distintos. Dentre as amostras obtidas foi possível identificar 14 *fingerprintings* com base no *hash* da imagem renderizada de modo *offscreen*, conforme mostra a Tabela 1.

Tabela 1. Entropia e unicidade por *fingerprinting*.

Fingerprint	Ocorrências	Entropia	Unicidade	Max e Min
cKEUdYviZNHQIOQ5zPWuNg	4	3.75	13.5	Entropia máxima: 5.75 Entropia mínima: 2.17 Unicidade máxima: 54 Unicidade mínima: 4.5
Nj9yFqmqIi3MKmAXs82Z6A	4	3.75	13.5	
8JaCw2dVtY-t_aEDh2o1Cg	3	4.17	18.0	
S_tZw9KaaBTlDRpx3mMA6g	12	2.17	4.5	
7W9-MjBUJCv1WlgAG_TFYQ	6	3.17	9.0	
60e9N960YjsE_Gh7Rh1E5A	1	5.75	54.0	
00fDOBxHczxSBC2wqT4oQg	2	4.75	27.0	
BnrIqoKCtbpWRE4sCFIj9w	12	2.17	4.5	
SvsxM9gg8Bbx5iWw8sfoew	2	4.75	27.0	
oOzLLixxJFfAmNmhmqmoEw	2	4.75	27.0	
00V5zGZwpdpaM3AwbLcAMg	1	5.75	54.0	
KJx9DLnkYhCBufkCrW8atg	2	4.75	27.0	
0zZ-iBzwBrqlElTC2Xy-RA	2	4.75	27.0	
m7tIb_-xU1QZTHCJm4-bnA	1	5.75	54.0	

Foi calculada a entropia e unicidade de cada *fingerprinting* com o objetivo de ter um critério de análise e poder avaliar a sua qualidade. Por exemplo, para o *fingerprinting* "***S_tZw9KaaBTlDRpx3mMA6g***" observa-se que este apresenta baixa entropia e baixa unicidade, o que indica que ele não é bom para esse grupo de dispositivos. Existem outros casos, como o "***60e9N960YjsE_Gh7Rh1E5A***", que apresenta uma alta entropia e alta unicidade, indicadores de um bom *fingerprinting*. Se considerarmos *fingerprintings*

bons somente aqueles com entropia maior ou igual a 4.75 e unicidade maior ou igual a 27.0, podemos observar que 57% estão nessa categoria, o que indica que o uso da técnica deste trabalho é boa e influência na qualidade de um bom *fingerprinting*. É importante mencionar que estas métricas melhoram quando atributos adicionais são utilizados, por exemplo o modelo do aparelho, os processadores, localização, etc.

6. Conclusão

Este trabalho apresentou a aplicabilidade de utilização de renderização de imagens aceleradas por GPU para *fingerprinting* de *smartphones*. Os resultados preliminares mostram que essa técnica é capaz de auxiliar na identificação de classes de dispositivos, principalmente se o *hash* da imagem gerada for utilizada em conjunto com outras informações do dispositivo.

Embora o uso de *fingerprinting* possa levantar questões relacionadas à privacidade do usuário, ele pode também ser explorado com o intuito de aumentar o nível de segurança em aplicações. Vale salientar que o método descrito neste trabalho não é capaz de identificar um usuário unicamente.

Como possibilidades de trabalhos futuros pode-se destacar: (I) investigar a geração de imagens com diferentes características, como modelos mais complexos, variações de posicionamento de câmera e luzes complexas; (II) reavaliar os resultados com base em uma quantidade de amostras mais significativa; (III) avaliar a estabilidade do *fingerprinting* em dispositivos recorrentes.

Referências

Cao, Y., Li, Song. and, Wijmans, E. (2017). “(Cross-)Browser Fingerprinting via OS and Hardware Level Features”.

Cheng, K. T., & Wang, Y. C. (2011, April). Using mobile GPU for general-purpose computing–a case study of face recognition on smartphones. In Proceedings of 2011 International Symposium on VLSI Design, Automation and Test (pp. 1-4). IEEE.

Eckersley, P. (2010). "How unique is your web browser?". In Proceedings of the 10th international conference on Privacy enhancing technologies (PETS'10), Mikhail J. Atallah and Nicholas J. Hopper (Eds.). Springer-Verlag, Berlin, Heidelberg, 1-18.

Ferreira T., Christof and Jonker, H. (2018). “Investigating Fingerprinters and Fingerprinting- alike Behaviour of Android Applications”. Proceedings on European Symposium on Research in Computer Security (ESORICS).

Kurtz, A., Gascon H., Becker, T., Rieck, K., F. Felix. (2016) “Fingerprinting Mobile Devices Using Personalized Configurations”. Proceedings on Privacy Enhancing Technologies (PoPETS) (1), 4–19.

Laperdrix P., Rudametkin W. and Baudry B., (2016). "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints," *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, pp. 878-894. doi: 10.1109/SP.2016.57

Mowery, K., & Shacham, H. (2012). "Pixel Perfect : Fingerprinting Canvas in HTML5". In *Proceedings of W2SP*, pages 1-12, IEEE.

Wu W., Wu J., Wang Y.; Ling Z.; and Yang M. (2016). “Efficient Fingerprinting-Based Android Device Identification With Zero-Permission Identifiers”. IEEE Access. PP. 1-1. doi: 10.1109/ACCESS.2016.2626395.