ARMageddon: Cache Attacks on Mobile Devices

Acadêmico: Eduardo Alberto Schmoller

Autores: Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice e Stefan Mangard. **Graz University of Technology, Austria, 25th USENIX Security Symposium**

Pato Branco, 22 de Junho de 2017

Objetivos

Demonstrar técnicas de ataque à memórias cache e a aplicação dessas técnicas em smartphones android com processadores ARM.

Obtenção de dados de aplicativos sem a necessidade de permissões de superusuário (*root*).

Método de comunicação entre processos.

Ataque a algoritmos criptográficos, detecção de toques, arastos na tela e tamanho de palavras digitadas.

Cache - Arm

- Velocidade de acesso.
- Politica de Substituição
  - LRU
  - Pseudo-Random
- Cache Inclusivo ou Não-Inclusivo
- Cache L2 Compartilhado → Core pode alterar conteúdo
- Mapeamento memória principal → Cache

Desafios

1. Cache L2 não inclusivo em dispositivos ARM.
2. Multiplas CPU não compartilham o Cache.
3. CPUs ARM sem suporte a instrução *flush.*
4. Pseudo-Ramdom como politica de substituição cache.
5. Obtenção de medidas de tempo com alta precisão.

Técnicas

- Prime+Probe
- Flush+Reload
- Evict+Reload
- Flush+Flush

Dispositivos Utilizados

Table 1: Test devices used in this paper.

Device	SoC	CPU (cores)	L1 caches	L2 cache	Inclusiveness
OnePlus One	Qualcomm Snapdragon 801	Krait 400 (2) 2.5 GHz	2× 16 KB, 4-way, 64 sets	2 048 KB, 8-way, 2 048 sets	non-inclusive
Alcatel One Touch Pop 2	Qualcomm Snapdragon 410	Cortex-A53 (4) 1.2 GHz	4× 32 KB, 4-way, 128 sets	512 KB, 16-way, 512 sets	instruction-inclusive, data-non-inclusive
Samsung Galaxy S6	Samsung Exynos 7 Octa 7420	Cortex-A53 (4) 1.5 GHz Cortex-A57 (4) 2.1 GHz	4× 32 KB, 4-way, 128 sets 4× 32 KB, 2-way, 256 sets	256 KB, 16-way, 256 sets 2 048 KB, 16-way, 2 048 sets	instruction-inclusive, data-non-inclusive instruction-non-inclusive, data-inclusive

Ataque Cache

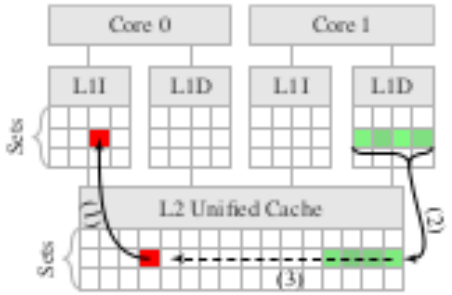

Figure 1: Cross-core instruction cache eviction through data accesses.

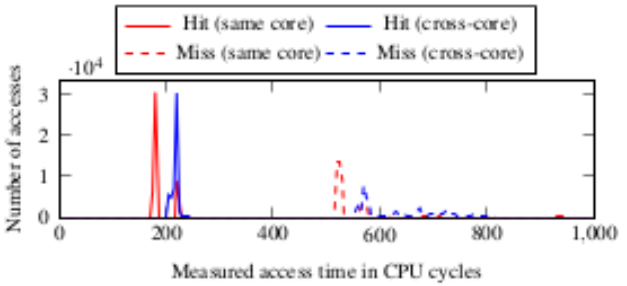

Figure 2: Histograms of cache hits and cache misses measured same-core and cross-core on the OnePlus One.

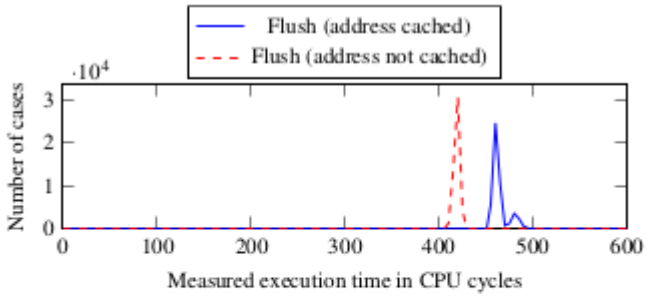

Figure 3: Histograms of the execution time of the flush operation on cached and not cached addresses measured on the Samsung Galaxy S6.

Fast Cache Eviction

Mais de 4200 combinações testadas.

Table 2: Different eviction strategies on the Krait 400.

N	*A*	*D*	Cycles	Eviction rate
-	-	-	549	100.00%
11	2	2	1 578	100.00%
12	1	3	2 094	100.00%
13	1	5	2 213	100.00%
16	1	1	3 026	100.00%
24	1	1	4 371	100.00%
13	1	2	2 372	99.58%
11	1	3	1 608	80.94%
11	4	1	1 948	58.93%
10	2	2	1 275	51.12%

Table 3: Different eviction strategies on the Cortex-A53.

N	*A*	*D*	Cycles	Eviction rate
-	-	-	767	100.00%
23	2	5	6 209	100.00%
23	4	6	16 912	100.00%
22	1	6	5 101	99.99%
21	1	6	4 275	99.93%
20	4	6	13 265	99.44%
800	1	1	142 876	99.10%
200	1	1	33 110	96.04%
100	1	1	15 493	89.77%
48	1	1	6 517	70.78%

Medidas de Tempo

- Unprivileged syscall → PERF_COUNT_HW_CPU_CYCLES
- POSIX function → clock_gettime()
- Dedicated thread timer → Contador incrementando uma variável global.

Medidas de Tempo

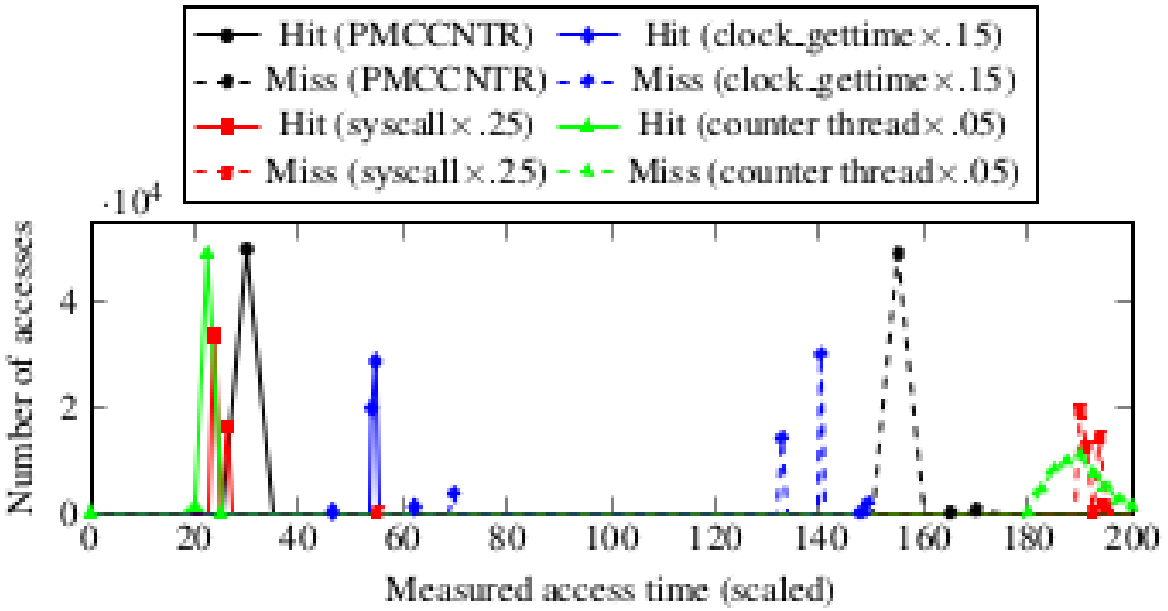

Figure 4: Histogram of cross-core cache hits/misses on the Alcatel One Touch Pop 2 using different methods. X-values are scaled for visual representation.

High Performance Covert Channels

Mecanismo de comunicação entre dois processos sem utilizar recursos do SO.

1. Definir (*n* + *s* + *c)* endereços em alguma biblioteca.

2. Cada endereço representa um *bit* na comunicação.

3. Medida do tempo de acesso, se estiver em cache um *bit* com valor 1 foi transmitido.

High Performance Covert Channels

Table 4: Comparison of covert channels on Android.

Work	Type	Bandwidth [bps]	Error rate
Ours (Samsung Galaxy S6)	*Flush+Reload*, cross-core	**1 140 650**	1.10 %
Ours (Samsung Galaxy S6)	*Flush+Reload*, cross-CPU	**257 509**	1.83 %
Ours (Samsung Galaxy S6)	*Flush+Flush*, cross-core	**178 292**	0.48 %
Ours (Alcatel One Touch Pop 2)	*Evict+Reload*, cross-core	**13 618**	3.79 %
Ours (OnePlus One)	*Evict+Reload*, cross-core	**12 537**	5.00 %
Marforio et al. [36]	Type of Intents	4 300	–
Marforio et al. [36]	UNIX socket discovery	2 600	–
Schlegel et al. [48]	File locks	685	–
Schlegel et al. [48]	Volume settings	150	–
Schlegel et al. [48]	Vibration settings	87	–

Attacking a Shared Library

Mapeamento de um mesmo endereço de memória em varios espaços de memória cache.

Cada endereço da biblioteca representa uma ação.

Android < 4.4 **Dalvik VM**, sem bibliotecas compartilhadas.

Android > 4.4 **ART,** Binários compilados são compartilhados.

Attacking User Input on Smartphones

Mapear bibliotecas compartilhadas. Ex. *libinput.so*

Gerar eventos de entradas (*key, longpress, tap, swipe, text*) e medir o tempo de resposta de determindos endereços.

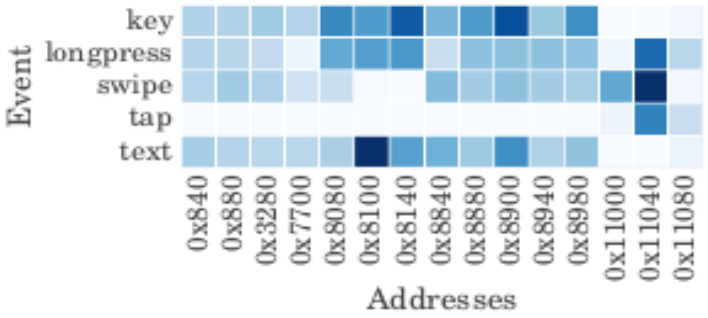

Figure 5: Cache template matrix for `libinput.so`.

Attacking User Input on Smartphones

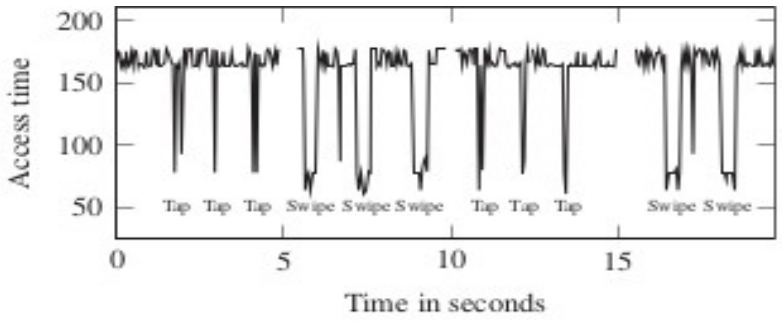

Figure 6: Monitoring address `0x11040` of `libinput.so` on the Alcatel One Touch Pop 2 reveals taps and swipes.

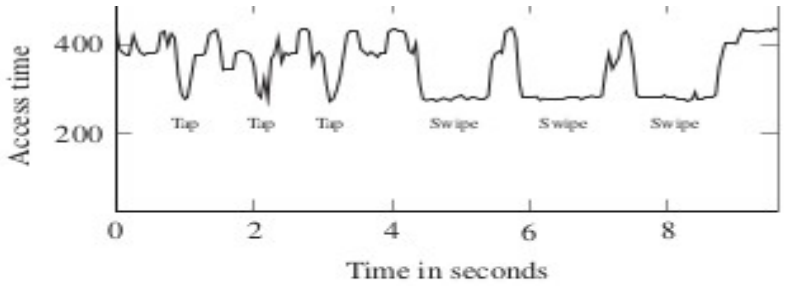

Figure 7: Monitoring address `0xDC5C` of `libinput.so` on the Samsung Galaxy S6 reveals tap and swipe events.

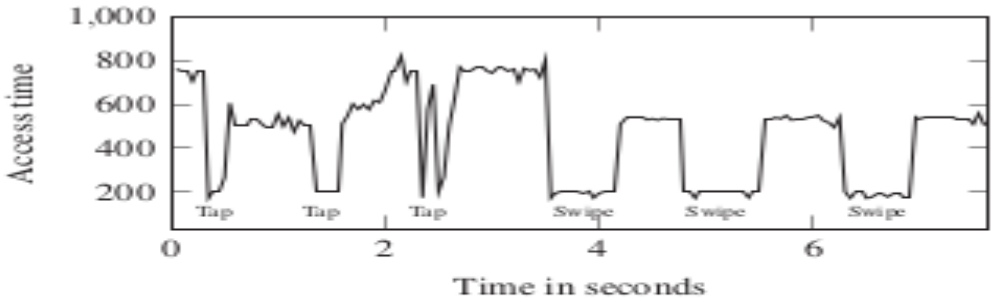

Figure 8: Monitoring address `0xBFF4` of `libinput.so` on the OnePlus One reveals tap and swipe events.

Attacking ART Binaries

ART é pré-compilado diferente do Java VM.

Mapeamento do ART do teclado AOSP para detecção do tamanho de palavras.

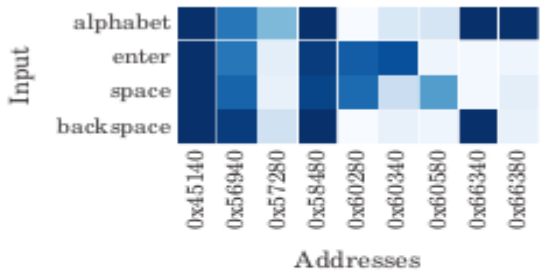

Figure 9: Cache template matrix for the default AOSP keyboard.

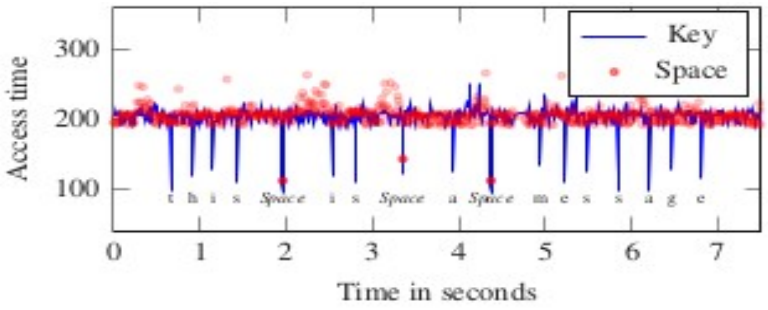

Figure 10: *Evict+Reload* on 2 addresses in `custpack@app@withoutlibs@LatinIME.apk@classes.dex` on the Alcatel One Touch Pop 2 while entering the sentence "this is a message".

ART Binaries / Shared Library

Permite que aplicativos nocivos obter dados do usuário.

- GPS, bluetooth, audio, câmera, NFC, WEB, sensores em geral.

Sem necessitar de permissão de superusuário aplicativo consegue mapear e salvar qualquer atividade.

Websites rodando JavaScript conseguem acessar essas informações

Attack on Cryptographic Algorithms

Ataque ao AES T-Table, Bouncy Castle, OpenSSL < 1.0.1

AES T-Table contêm valores pré computados de transformações, permitindo criptografias apenas com XOR: $si = pi \oplus ki$

Sabendo pi (valor a ser criptografado) e o endereço da tabela usado descobre-se a chave usada: $ki = si \oplus pi$

Monitorando apenas 1 endereço a cada criptografia, em 3207 processos.

Medidas preventivas

- Restringir o acesso a */proc/self/pagemap*
- Restringir o acesso a */proc/pid/pagemap*
- Restringir o acesso a */proc/pid/maps*
- Acesso ao ART e dados de todos os aplicativos */data/dalvik-cache/*
- Utilizar istruções AES em hardware para criptografia.
- Março de 2016 Google limita o acesso aos diretórios */proc/pid/pagemap*

Conclusão

Trabalho demonstrou possibilidade e brechas na arquitetura ARM.

Alta precisão em reconhecer *taps* e *swipes* na tela.

Demostrado um método eficaz para ataques em soluções de criptografias.

Referências

- *GRUSS , D., MAURICE , C., AND MANGARD , S. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In DIMVA'16 (2016).*
- *GRUSS , D., MAURICE , C., WAGNER , K., AND MANGARD , S. Flush+Flush: A Fast and Stealthy Cache Attack. In DIMVA'16 (2016).*
- *LIPP , D., GRUSS , D., SPREITZER , R., MAURICE , C., MANGARD , S. ARMageddon: Cache Attacks on Mobile Devices. In 25th USENIX Security Symposium.*
- *OSVIK, D., SHAMIR, A., TROM, E. Cache Attacks and Countermeasures: The Case of AES. In CT-RSA 2006, pp. 1-20.*

ARMageddon: Cache Attacks on Mobile Devices

Acadêmico: Eduardo Alberto Schmoller

Autores: Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice e Stefan Mangard. **Graz University of Technology, Austria, 25th USENIX Security Symposium**

Pato Branco, 22 de Junho de 2017