Tappan Zee (North) Bridge

Mining Memory Accesses for Introspection

Brendan Dolan-Gavitt, Tim Leek, Josh Hodosh, and Wenke Lee

ACM CCS 11/6/2013

This work is sponsored by the Assistant Secretary of Defense for Research & Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Photo by joseph a

Background: Introspection

- Often we want to provide isolation for a security-sensitive process
- This isolation limits visibility: a *semantic gap* that must be bridged
- Lots of existing work on automatically bridging this gap: Virtuoso, VMST

Motivation

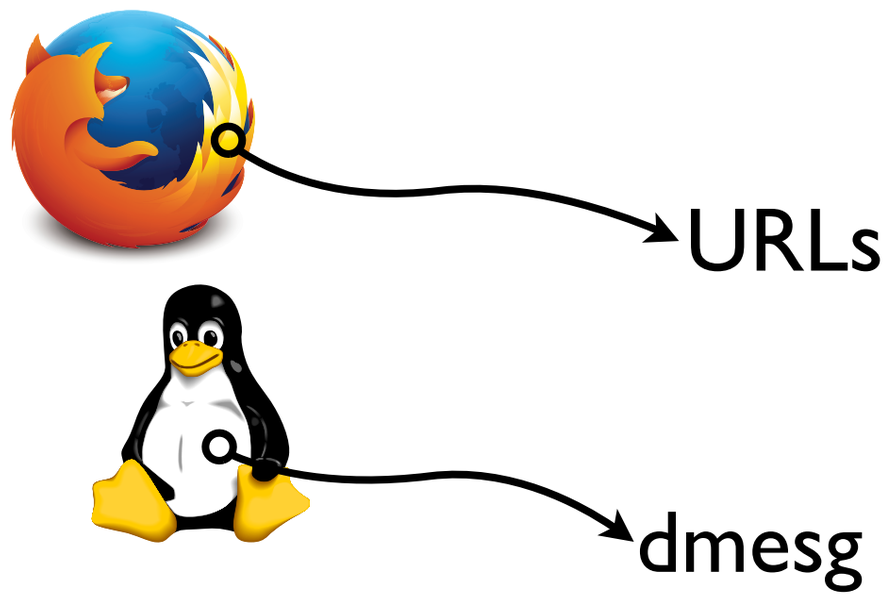
- Application introspection is hard
- Finding places to hook: also hard
- *Neither scenario is supported with existing techniques*
  - Currently done by *manual* reverse engineering

Fond Memories

- We focus on a primitive operation used by (nearly) all apps: memory access
- *Tapping* memory accesses provides a way to find places to hook for OS and application introspection
- Memory accesses provide *streams* of data that illuminate the inner workings of apps

Tap Points

- Abstraction for "memory access made at point in program"
- Idea: Data at a tap point of same "type"
- Consists of:
  - Caller
  - Program Counter
  - "Program" (CR3)



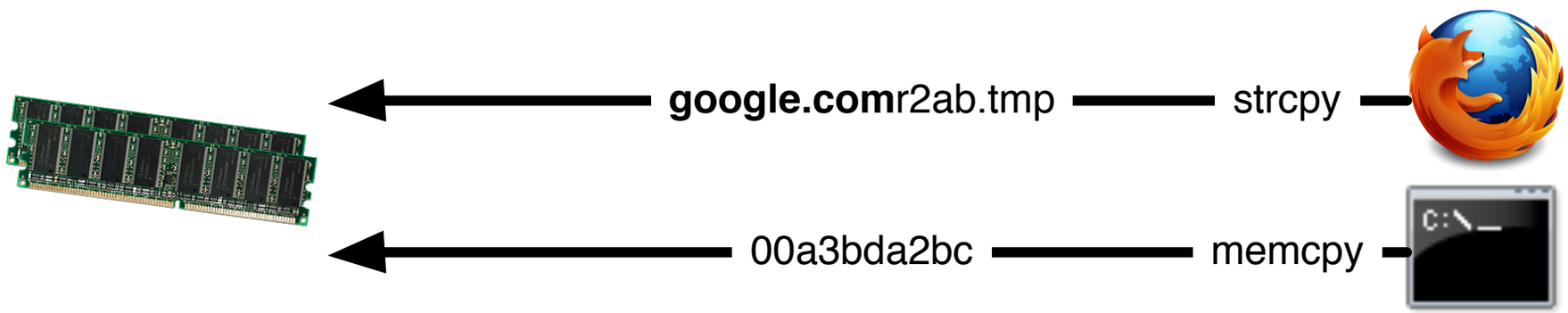
Context

(a)

Context


00a3bdgoogle.comr2ab.tmpa2bc

(a)

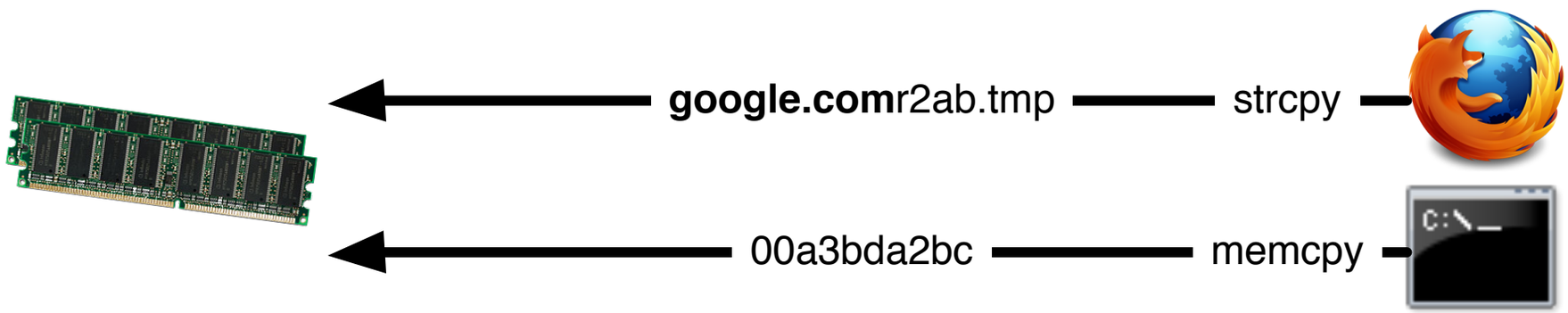

google.comr2ab.tmp
strcpy
00a3bda2bc
memcpy
C:_

(b)

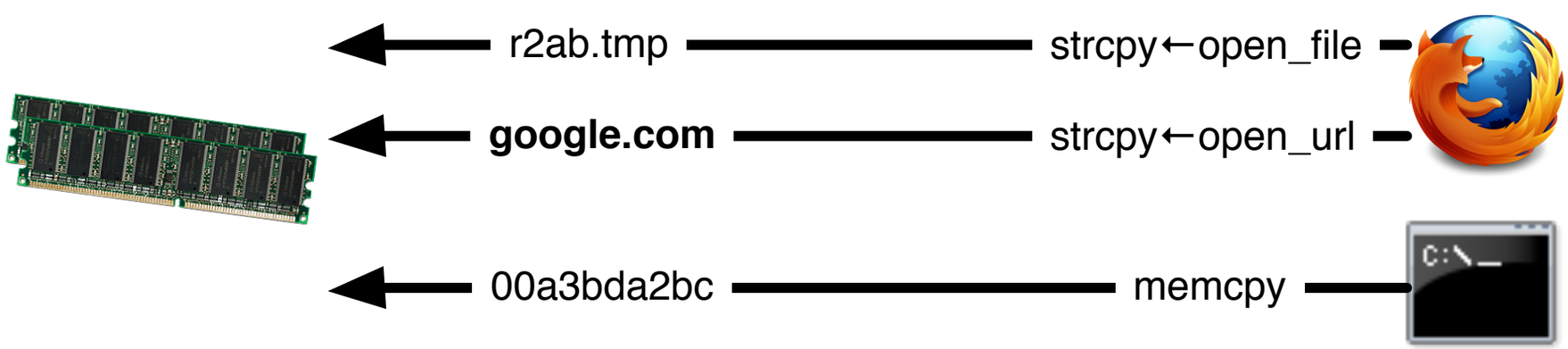
Context


00a3bd**google.com**r2ab.tmpa2bc

(a)


google.comr2ab.tmp
strcpy
00a3bda2bc
memcpy

(b)


r2ab.tmp
strcpy←open_file
google.com
strcpy←open_url
00a3bda2bc
memcpy

(c)

Sample Tap Points

Content

Read Write

Tap

Code

```
00646517 0064A423 Kernel   0064A423 push    ebx
00646517 0064A424 Kernel   0064A424 push    [ebp+var_28]
00646517 0064A427 Kernel   0064A427 push    esi
00646517 0064A428 Kernel   0064A428 call    _memcpy

                           _memcpy:
                           [...]
                           00430E08 shr     ecx, 2
                           00430E0B and     edx, 3
                           00430E0E cmp     ecx, 8
                           00430E11 jb      short loc_430E3C
0064A42D 00430E13 Kernel   00430E13 rep movsd
0064A42D 00430E15 Kernel   00430E15 jmp     off_430F2C[edx*4]
```

Sample Tap Points

Content

Read Write

Tap

Code

```
00FFABED
```

```
00646517 0064A423 Kernel   0064A423 push    ebx
00646517 0064A424 Kernel   0064A424 push    [ebp+var_28]
00646517 0064A427 Kernel   0064A427 push    esi
00646517 0064A428 Kernel   0064A428 call    _memcpy

                           _memcpy:
                           [...]
                           00430E08 shr     ecx, 2
                           00430E0B and     edx, 3
                           00430E0E cmp     ecx, 8
                           00430E11 jb      short loc_430E3C
0064A42D 00430E13 Kernel   00430E13 rep movsd
0064A42D 00430E15 Kernel   00430E15 jmp     off_430F2C[edx*4]
```

Sample Tap Points

Content		Tap			Code	
Read	Write					
	00FFABED	00646517	0064A423	Kernel	0064A423	push ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push [ebp+var_28]
		00646517	0064A427	Kernel	0064A427	push esi
		00646517	0064A428	Kernel	0064A428	call _memcpy
						_memcpy:
						[...]
					00430E08	shr ecx, 2
					00430E0B	and edx, 3
					00430E0E	cmp ecx, 8
					00430E11	jb short loc_430E3C
		0064A42D	00430E13	Kernel	00430E13	rep movsd
		0064A42D	00430E15	Kernel	00430E15	jmp off_430F2C[edx*4]

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
		00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
		0064A42D	00430E13	Kernel	00430E13	rep movsd	
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy

```
                                    _memcpy:
                                    [...]
                                    00430E08 shr     ecx, 2
                                    00430E0B and     edx, 3
                                    00430E0E cmp     ecx, 8
                                    00430E11 jb      short loc_430E3C
0064A42D 00430E13 Kernel    00430E13 rep movsd
0064A42D 00430E15 Kernel    00430E15 jmp     off_430F2C[edx*4]
```

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Dev	\Dev	0064A42D	00430E13	Kernel	00430E13	rep	movsd
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\	\Device\	0064A42D	00430E13	Kernel	00430E13	rep	movsd
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content

Read	Write
	00FFABED
00123456	00123456
	00ABCDEF
	0064A42D
\Device\Hard	\Device\Hard

Tap

```
00646517 0064A423 Kernel
00646517 0064A424 Kernel
00646517 0064A427 Kernel
00646517 0064A428 Kernel
0064A42D 00430E13 Kernel
0064A42D 00430E15 Kernel
```

Code

```
0064A423 push    ebx
0064A424 push    [ebp+var_28]
0064A427 push    esi
0064A428 call    _memcpy

_memcpy:
[...]
00430E08 shr     ecx, 2
00430E0B and     edx, 3
00430E0E cmp     ecx, 8
00430E11 jb      short loc_430E3C
00430E13 rep movsd
00430E15 jmp     off_430F2C[edx*4]
```

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\Harddisk	\Device\Harddisk	0064A42D	00430E13	Kernel	00430E13	rep	movsd
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content		Tap			Code		
Read	Write						
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\Harddisk	\Device\Harddisk	0064A42D	00430E13	Kernel	00430E13	rep movsd	
00430F3C		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Finding Tap Points

- Millions of tap points in even short slices of system execution
- Gigabytes of data generated
- Need effective, efficient ways of finding useful tap points

Monitoring Memory Accesses

- Can instrument QEMU to log all memory accesses – SLOW
- To get around this we use *whole-system record and replay*
- Implemented in our open source dynamic analysis framework (PANDA)

General Strategy

- Identify behavior of interest ("URL access")
- Training: Create recording in which behavior occurs ("visit google.com")
- Search: Replay recording with instrumentation and look for tap point with desired content

Search Strategies

- Known knowns
- Known unknowns
- Unknown unknowns

Known Knowns: String Searching

- Can be efficiently implemented using one counter per tap point per search string
- Millions of tap points can be searched with a few megabytes of memory
- We found: URLs, filenames, window titles, SSL/TLS master keys

Known Knowns: TLS/SSL Keys

- If exact key is not known in advance (e.g., malicious server) we cannot string search
- However, we can do trial decryption on all 48-byte strings seen at tap points
- TLS MAC allows us to verify decryption

sslmalware.pcap [Wireshark 1.8.0rc2 (SVN Rev 43337 from /trunk-1.8)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: ip.addr== 69.36.10.150 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
941	781.730265	69.36.10.150	10.0.2.15	TCP	54	https > 49267 [ACK] Seq=1 Ack=169 Win=8760 Len=0
942	781.764680	69.36.10.150	10.0.2.15	TLSv1	1201	Server Hello, Certificate, Server Hello Done
943	781.809998	10.0.2.15	69.36.10.150	TLSv1	252	Client Key Exchange, Change Cipher Spec, Finished
944	781.835306	69.36.10.150	10.0.2.15	TCP	54	https > 49267 [ACK] Seq=1148 Ack=367 Win=8760 Len=0
945	781.851088	69.36.10.150	10.0.2.15	TLSv1	113	Change Cipher Spec, Finished
946	781.914346	10.0.2.15	69.36.10.150	HTTP	603	GET /putallow.asp?type=pthi.tmp&hostname=QEMU-RR-10
947	781.939180	69.36.10.150	10.0.2.15	TCP	54	https > 49267 [ACK] Seq=1207 Ack=916 Win=8760 Len=0
948	781.951902	69.36.10.150	10.0.2.15	HTTP	267	HTTP/1.1 200 OK
949	781.990646	10.0.2.15	69.36.10.150	TCP	54	49226 > https [FIN, ACK] Seq=1385 Ack=1974 Win=638
950	781.990676	69.36.10.150	10.0.2.15	TCP	54	https > 49226 [ACK] Seq=1974 Ack=1386 Win=8760 Len=
951	782.010808	10.0.2.15	69.36.10.150	TCP	66	49268 > https [SYN] Seq=0 Win=8192 Len=0 MSS=1460 W
952	782.022046	69.36.10.150	10.0.2.15	TCP	54	https > 49226 [FIN, ACK] Seq=1974 Ack=1386 Win=8760
953	782.023247	10.0.2.15	69.36.10.150	TCP	54	49226 > https [ACK] Seq=1386 Ack=1975 Win=63814 Len

```
[Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 49267 (49267), Dst Port: https (443), Seq: 367, Ack: 1207, Len: 549
Secure Sockets Layer
Hypertext Transfer Protocol
  GET /putallow.asp?type=pthi.tmp&hostname=QEMU-RR-10.0.2.15-hi1031m HTTP/1.1\r\n
  Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powe
  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.7) Gecko/2009021910 Firefox/3.0.7\r\n
  Host: www.hi-tecsolutions.org\r\n
  Connection: Keep-Alive\r\n
  Cache-Control: no-cache\r\n
```

```
0000  47 45 54 20 2f 70 75 74  61 6c 6c 6f 77 2e 61 73   GET /put allow.as
0010  70 3f 74 79 70 65 3d 70  74 68 69 2e 74 6d 70 26   p?type=p thi.tmp&
0020  68 6f 73 74 6e 61 6d 65  3d 51 45 4d 55 2d 52 52   hostname =QEMU-RR
0030  2d 31 30 2e 30 2e 32 2e  31 35 2d 68 69 31 30 33   -10.0.2. 15-hi103
0040  31 6d 20 48 54 54 50 2f  31 2e 31 0d 0a 41 63 63   1m HTTP/ 1.1..Acc
0050  65 70 74 3a 20 69 6d 61  67 65 2f 67 69 66 2c 20   ept: ima ge/gif,
0060  69 6d 61 67 65 2f 78 2d  78 62 69 74 6d 61 70 2c   image/x- xbitmap,
0070  20 69 6d 61 67 65 2f 6a  70 65 67 2c 20 69 6d 61    image/j peg, ima
0080  67 65 2f 70 6a 70 65 67  2c 20 61 70 70 6c 69 63   ge/pjpeg , applic
```

Frame (603 bytes) Decrypted SSL data (508 bytes)

File: "/Users/moyix/sslmalware.pcap" 706 KB 00:15:26 Packets: 132... Profile: Default

Known Unknowns: Information Retrieval

- Given some training examples, find tap points containing "similar" data
- We compute bigram byte statistics for each tap point & training examples
- Sort by Jensen-Shannon divergence (similar to mutual information / Kullback–Leibler divergence)

$$JSD(P,Q) = H\left(\frac{P+Q}{2}\right) - \frac{H(P)+H(Q)}{2}$$

Finding dmesg (training)

```
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.2.0-3-amd64 (Debian 3.2.23-1) (debian-kernel@lists.debian.org) (gcc version 4.6.3 (Debian 4.6.3-8) ) #1 SMP Mon Jul 23 02:45:17 UTC 2012
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.2.0-3-amd64 root=UUID=88af2518-864f-486a-822b-922675753f91 ro quiet
[    0.000000] BIOS-provided physical RAM map:
[    0.000000]  BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
[    0.000000]  BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
[    0.000000]  BIOS-e820: 00000000000ca000 - 00000000000cc000 (reserved)
[    0.000000]  BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
[    0.000000]  BIOS-e820: 0000000000100000 - 00000000bfef0000 (usable)
[    0.000000]  BIOS-e820: 00000000bfef0000 - 00000000bfeff000 (ACPI data)
[    0.000000]  BIOS-e820: 00000000bfeff000 - 00000000bff00000 (ACPI NVS)
[    0.000000]  BIOS-e820: 00000000bff00000 - 00000000c0000000 (usable)
[    0.000000]  BIOS-e820: 00000000e0000000 - 00000000f0000000 (reserved)
[    0.000000]  BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
[    0.000000]  BIOS-e820: 00000000fee00000 - 00000000fee01000 (reserved)
[    0.000000]  BIOS-e820: 00000000fffe0000 - 0000000100000000 (reserved)
[    0.000000]  BIOS-e820: 0000000100000000 - 0000000440000000 (usable)
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] DMI present.
[    0.000000] DMI: VMware, Inc. VMware Virtual Platform/440BX Desktop Reference Platform, BIOS 6.00 02/22/2012
[    0.000000] Hypervisor detected: VMware
[    0.000000] e820 update range: 0000000000000000 - 0000000000010000 (usable) ==> (reserved)
[    0.000000] e820 remove range: 00000000000a0000 - 0000000000100000 (usable)
```

dmesg

```
Copyright (c) 1992-2012 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 9.0-RELEASE #0: Tue Jan  3 07:15:25 UTC 2012
    root@obrian.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC i386
CPU: QEMU Virtual CPU version 1.0,1 (2300.80-MHz 686-class CPU)
  Origin = "AuthenticAMD"  Id = 0x623  Family = 6  Model = 2  Stepping = 3
  Features=0x783fbfd<FPU,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,MMX,FXSR,SSE,SSE2>
  Features2=0x80802001<SSE3,CX16,POPCNT,HV>
  AMD Features=0x20100800<SYSCALL,NX,LM>
  AMD Features2=0x65<LAHF,SVM,ABM,SSE4A>
```

FreeBSD

```
APIC disabled, using legacy PIC

MINIX 3.2.0. (116fcea)
Copyright 2012, Vrije Universiteit, Amsterdam, The Netherlands
MINIX is open source software, see http://www.minix3.org
Initiating legacy i8253 timer
CPU 0 freq 2192 MHz
Started VFS: 8 worker thread(s)
Thu Apr  4 19:09:26 GMT 2013
e1000#0: Intel PRO/1000 MT Desktop Adapter (8086/100e/00) at 0.3.0
```

MINIX

```
Using mode 0x118
VESA compatible graphics!
Welcome to the Haiku boot loader!
number of drives: 1
add_partitions_for(0x001051cc, mountFS = no)
add_partitions_for(fd = 0, mountFS = no)
0x00105320 Partition::Partition
0x00105320 Partition::Scan()
check for partitioning_system: EFI GUID Partition Map
check for partitioning_system: Intel Partition Map
  priority: 810
```

Haiku

```
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 2.6.32-5-amd64 (Debian 2.6.32-46) (dannf@debian.org) (gcc version 4.3.5 (Debian 4.3.5-4) ) #1 SMP Sun Sep 23 10:07:46 UTC 2012
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-2.6.32-5-amd64 root=UUID=add45803-2be7-4f6c-a13f-bab8f3ded507 ro quiet
```

Linux

Unknown Unknowns: Clustering

- Group tap points containing "similar" data together
- Algorithm: *K*-means with Jensen-Shannon as distance metric
- Unsupervised learning – no training

FreeBSD Boot Cluster

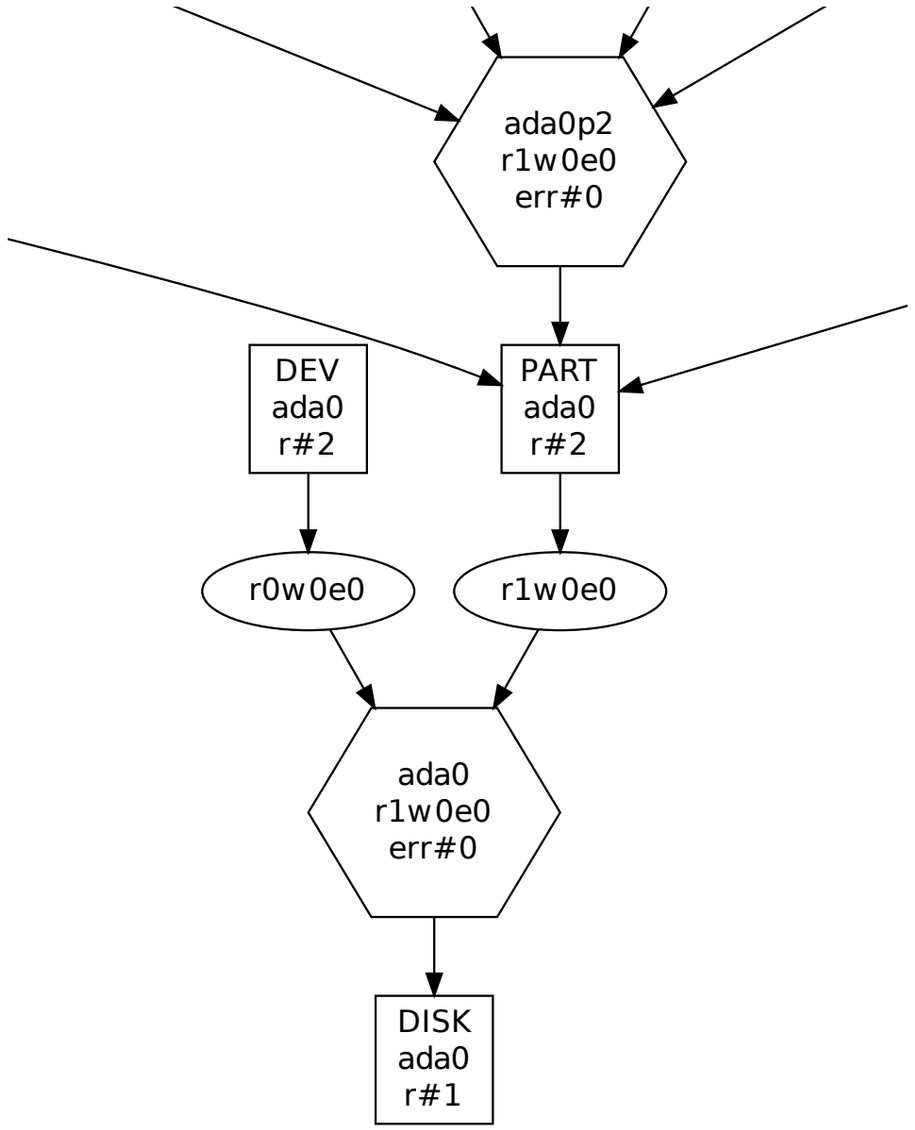
```
/faa_N_=_peOfA=fA=feTr=tul.n=_eo/.b_Yt_vtectvifat=a=-sd_Ee
Ofu=u_Oy:nF:tRseeeeEfciOtmdtuinlrlrrlpp/nppfpcepinl=l=.llN
lNlgllpl_.4l_l_2/l_l_22lileldlylo- 21laltlat=rrrsbgrskgni/

russian|Russian Users Accounts:  :charset=KOI8-R:        :
lang=ru_RU.KOI8-R:     :       :passwd_format=md5:     :co
pyright=/etc/COPYRIGHT:      :welcome=/etc/motd:     :sete
nv=MAIL=/var/mail/$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:

nss_compat.so.1dhclientShared object ‘‘nss_compat.so.1’’ n
ot found, required by ‘‘dhclient’’nss_nis.so.1dhclientShar
ed object ‘‘nss_nis.so.1’’ not found, required by ‘‘dhclie
nt’’nss_files.so.1dhclientShared object ‘‘nss_files.so.1’’

digraph geom {
z0xc1d8de00 [shape=box,label=’’PART\nada0\nr#2’’];
z0xc1f4f640 [label=’’r1w0e0’’];
z0xc1f4f640 -> z0xc1e9eb00;

/sbin/in/bin/sh/bin/stt/sbin/sysctl/bin/ps/sbin/sysctl/sbi
n/rcorde/bin/cat/sbin/md/sbin/sysctl/sbin/sysctl/bin/ken/s
bin/dumpon/bin/ln/bin/ps/sbin/sysctl/sbin/sysctl/sbin/sysc
tl/sbin/sysctl/bin/ps/bin/dd/sbin/sysctl/bin/dat/bin/df/sb
```



Clustering Evaluation

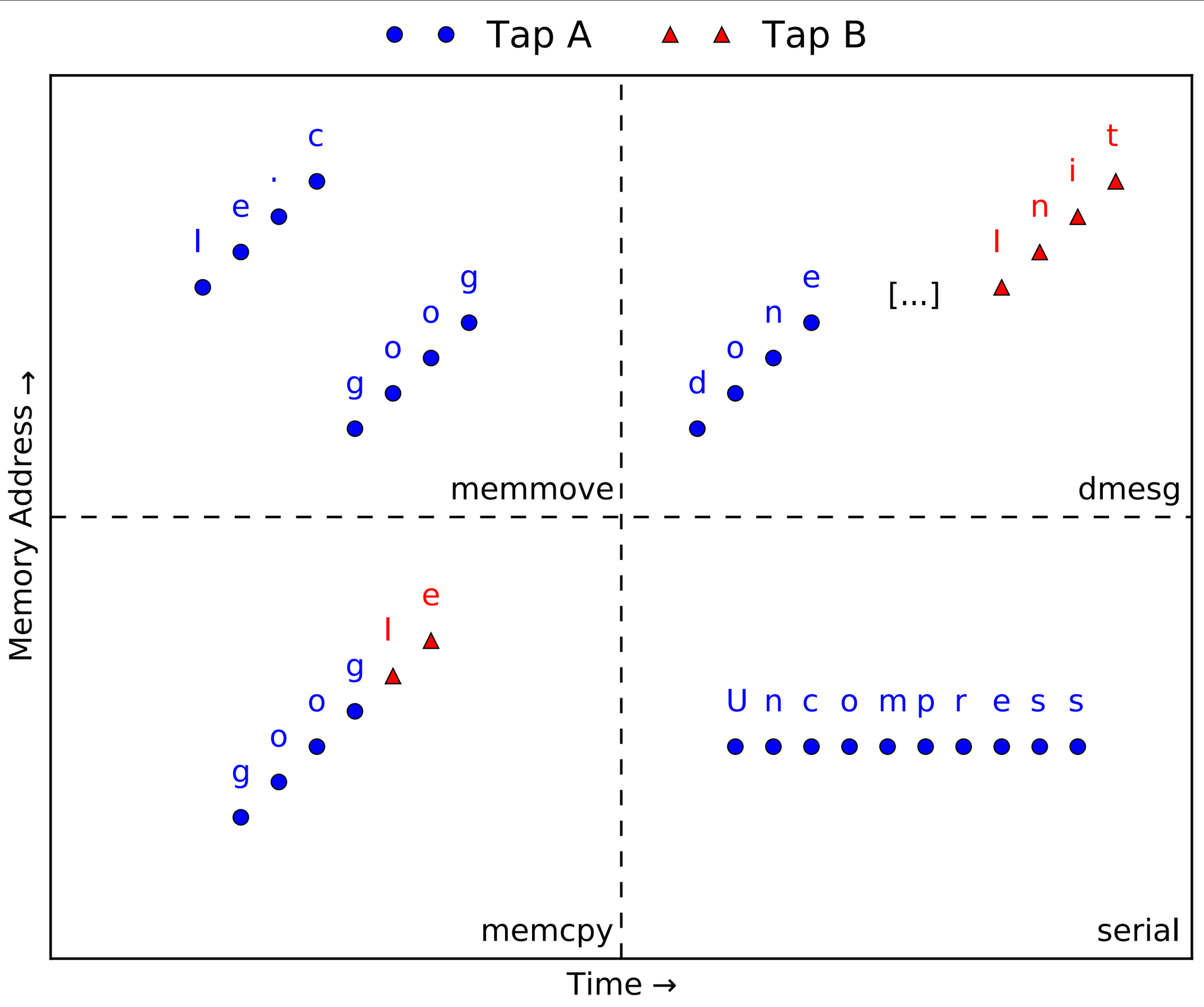
- Manually labelled ~3000 tap points
- Compared human labels to clustering
- Very poor agreement
  - Humans said most things were “binary pattern”

TZB is General

- Four introspection tasks: URL access, file access, SSL/TLS keys, boot messages
- Five operating systems: Windows 7, Linux, FreeBSD, Minix, and Haiku
- Two architectures: x86_64, ARM

Limitations

- Need to know data encoding
- One caller may not be enough (or too much!)
- Contents may be written (temporally) out of order


Tap A
Tap B
memmove
dmesg
memcpy
serial
Time →
Memory Address →
[...]
Uncompress

Future Work

- Clustering results suggestive, but more investigation is needed
- Application to memory forensics – online vs. offline
- Tap points in JITed code

Summary

- TZB locates *tap points*: places to interpose for extracting security-relevant information
- Analysis to *find* tap points is heavyweight, but is automated and only needs to be done once, offline
- Improves on state of the art: expensive and time-consuming manual reverse engineering

Thanks for Listening

- All code related to this project is available at https://github.com/moyix/panda
- Contact: brendan@cc.gatech.edu
  http://cc.gatech.edu/~brendan/
- Questions?

Performance

- Depends on analysis
- String search: ~10x
- SSL/TLS: ~500x