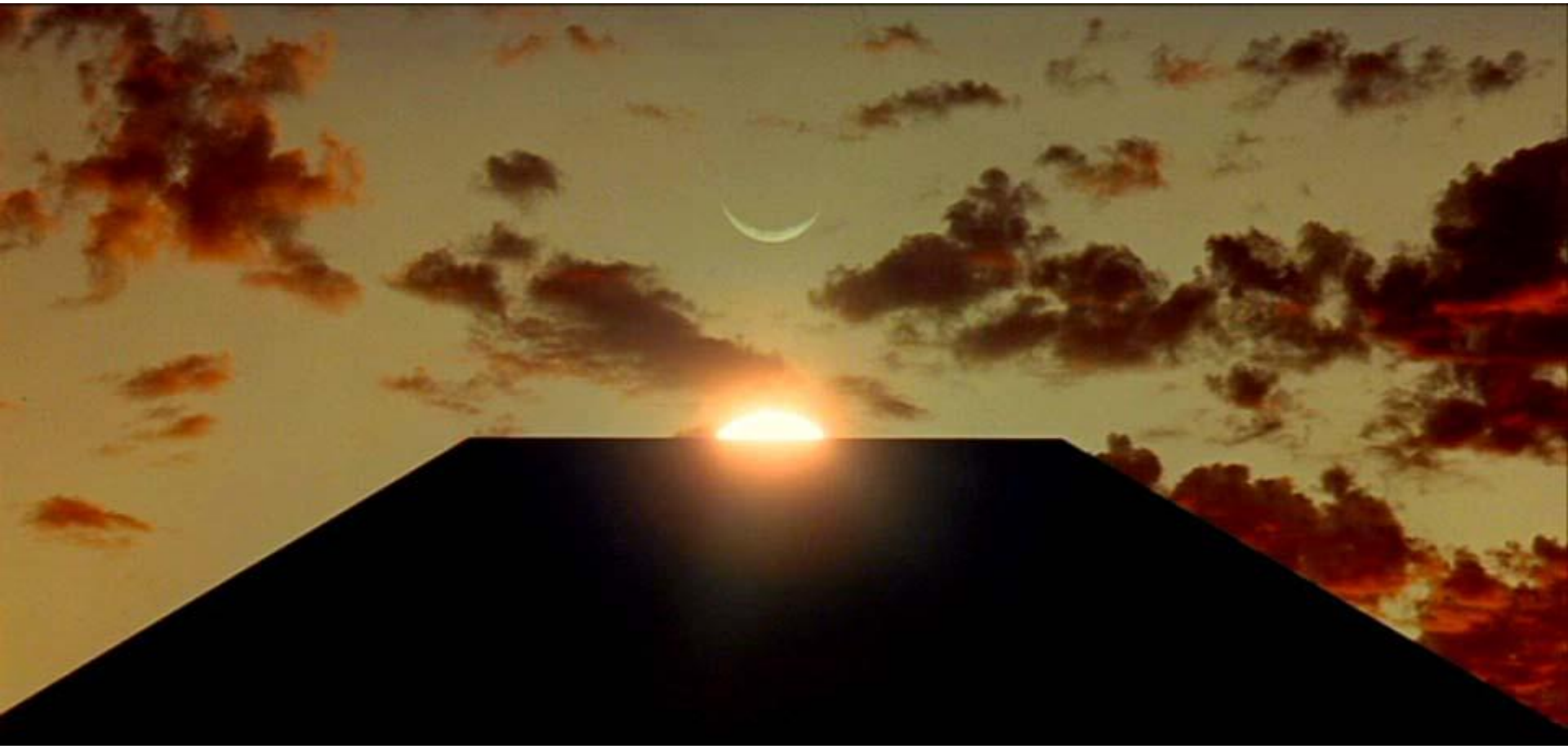
Chrome OS Hardening

http://outflux.net/slides/2012/bsides-pdx/chromeos.pdf

Security B-Sides PDX 2012

Kees Cook <keescook@google.com>
(pronounced "Case")

Overview



Overview

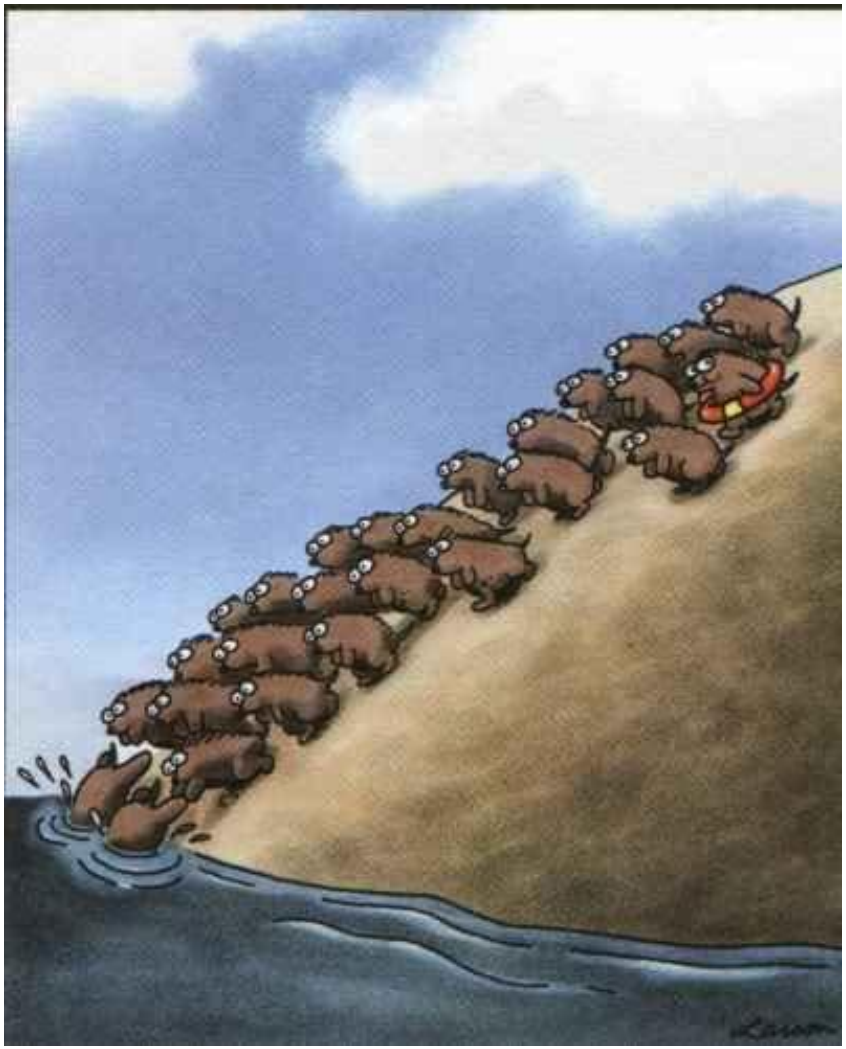
- Identifying the threat model
  - not the authenticated physically local user
- Doing what others do
  - common hardening strategies
- Doing what others will be doing
  - filling the gaps in common strategies
- Doing what others haven't done
  - Chrome OS is unique and so are our opportunities
- Testing ourselves
  - make sure this stuff actually works
- Seeing the future
  - what's next

Identifying the threat model

Identifying the threat model

- Malware
  - do not allow persistent control
- Theft
  - encrypt user and enterprise data
- Evil coffee shop patron
  - impose minimum physically local attack time
- Not the authenticated physically local user
  - developer mode

Doing what others do



Doing what others do

- Linux
  - TPM, GPT partitions, software auto-update
- Encryption
  - eCryptfs, dm-crypt
- Compiler hardening
  - stack-protector, FORTIFY_SOURCE, relro, bind-now, PIE
- Userspace hardening
  - RO/NX, ASLR, glibc runtime checks, namespaces, ptrace restrictions, link restrictions
- Kernel hardening
  - stack-protector, memory restriction, RO/NX, kptr_restrict

Linux: TPM

- Key storage
- Entropy source
- Data "sealing"

```
localhost ~ # tpm_version
  TPM 1.2 Version Info:
  Chip Version:        1.2.3.18
  Spec Level:          2
  Errata Revision:     2
  TPM Vendor ID:       IFX
  Vendor Specific data: 03120009 00
  TPM Version:         01010000
  Manufacturer Info:   49465800
localhost ~ # initctl stop tcsd
tcsd stop/waiting
localhost ~ # tpmc pcrread 0
865aedd337518e56f648440b81b4cbd9359fdff3
```

Linux: GPT partitions

STATE

KERN-A

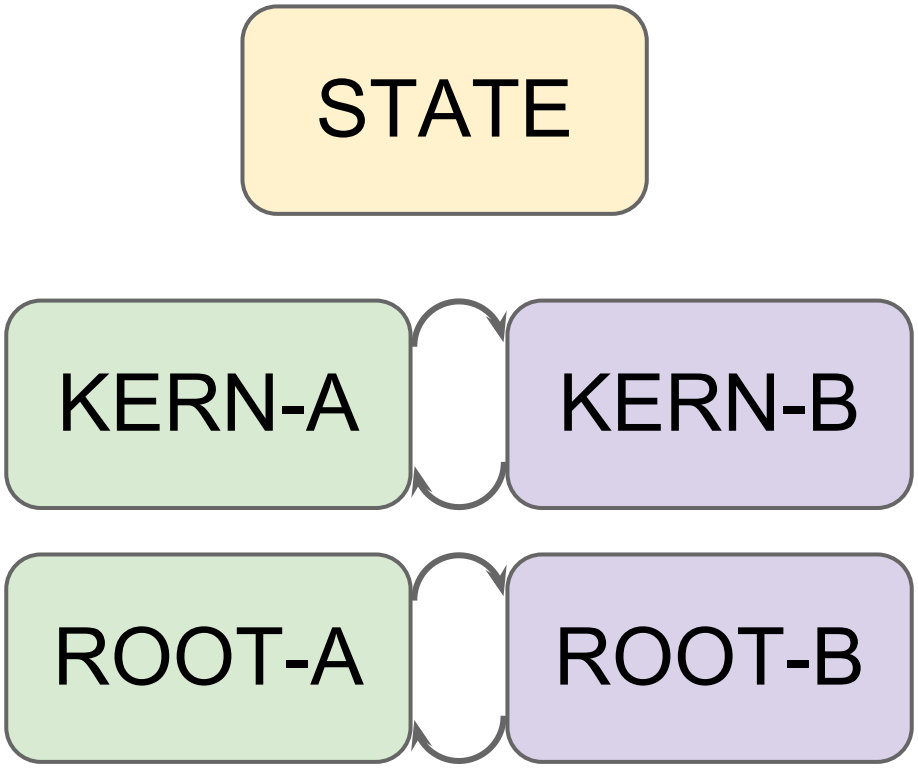
KERN-B

ROOT-A

ROOT-B

Linux: software auto-update

- Signed by Google
- Block-delta or full update



Encryption: eCryptfs

- Per-user, TPM-tied
- Mounted/unmounted at login/logout
- Overlay filesystem
- Need to share space between users

```
localhost ~ # cryptohome-path user keescook@chromium.org
/home/user/E1B39A9A13AA747E1F0D3EBD9708E21484616B1B
localhost ~ # mount | grep -i E1B39A9A
/home/.shadow/e1b.../vault on /home/.shadow/e1b.../mount type ecryptfs
(rw,nosuid,nodev,noexec,relatime,ecryptfs_sig=be...,
ecryptfs_fnek_sig=783...,ecryptfs_cipher=aes,ecryptfs_key_bytes=16,
ecryptfs_unlink_sigs)
/home/.shadow/e1b.../vault on /home/chronos/user ...
/home/.shadow/e1b.../vault on /home/user/E1B... ...
/home/.shadow/e1b.../vault on /home/root/E1B... ...
localhost ~ # ls -l /home/.shadow/e1b.../vault/user/
-rw-r--r--  1 chronos chronos        45056 Nov  7 14:35
ECRYPTFS_FNEK_ENCRYPTED.FXZtC.K15HueW-Sm351moE8t9.k00.
MnaSXZ43fRjGZnz6WGBMm4AX-QyyWWIeq19OYaiISZGRooJqg-
...
```

Encryption: dm-crypt

- Per-system, TPM-tied
- Mounted/unmounted at boot/shutdown
- Must share space: sparse backing file
  - discard (TRIM), hole-punching

```
localhost ~ # mount | grep enc
/dev/mapper/encstateful on /mnt/stateful_partition/encrypted type ext4 (rw,
nosuid,nodev,noexec,relatime,discard,commit=600,data=ordered)
/dev/mapper/encstateful on /var type ext4 (rw,nosuid,nodev,noexec,...
/dev/mapper/encstateful on /home/chronos type ext4 (rw,nosuid,nodev,...
localhost ~ # dmsetup table --showkeys /dev/mapper/encstateful
0 6748024 crypt aes-cbc-essiv:sha256 e9aef... 0 7:0 0 1 allow_discards
localhost ~ # dmsetup deps /dev/mapper/encstateful
1 dependencies  : (7, 0)
localhost ~ # ls -la /dev/loop0
brw-rw---- 1 root disk 7, 0 Nov  7 14:33 /dev/loop0
localhost ~ # losetup -a
/dev/loop0: [2049]:12 (/mnt/stateful_partition/encrypted.block)
```

Compiler: -fstack-protector

- Add canary
- Reorder variables

```
$ objdump -d -M intel exe...
```

prologue

```
mov     rax,QWORD PTR fs:0x28
mov     QWORD PTR [rbp-0x8],rax
```

...

epilogue

```
mov     rdx,QWORD PTR [rbp-0x8]
xor     rdx,QWORD PTR fs:0x28
je      400629 <func+0x5d>
call    4004a0 <__stack_chk_fail@plt>
...
```

BEFORE
function arguments ... saved program counter saved frame pointer local variables char foo[...]; int sensitive; char bar[...]; ...

AFTER
function arguments ... saved program counter saved frame pointer random canary local variables char foo[...]; char bar[...]; int sensitive; ...

Compiler: -D_FORTIFY_SOURCE=2

- Compile-time checking
  - return value usage (system(), write(), etc)
  - specification of optional arguments (open() modes)
  - when size of target buffers known: swap in bounded functions (sprintf -> snprintf)
  - when size of target buffers unknown: swap in runtime checking (strcpy -> __strcpy_chk)
- Runtime checking
  - block "%n" in format strings
  - block format string parameter skipping

```
localhost ~ # readelf -s /opt/google/chrome/chrome | grep _chk
  17: ... FUNC    GLOBAL DEFAULT  UND __printf_chk@GLIBC_2.3.4 (3)
  20: ... FUNC    GLOBAL DEFAULT  UND __vsnprintf_chk@GLIBC_2.3.4 (3)
  22: ... FUNC    GLOBAL DEFAULT  UND __strcat_chk@GLIBC_2.3.4 (3)
...
```

Userspace: RO/NX

```
localhost ~ # cat boom.c
#include <stdlib.h>
unsigned int ret_insns = 0xc3c3c3c3;
void main(int argc, char *argv[])
{
    void(*func)(void) = (void*)strtoul(argv[1], NULL, 16);
    func();
}
localhost ~ # cc boom.c -o boom
localhost ~ # readelf -l boom | egrep -A1 'LOAD|VirtAddr'
  Type  Offset             VirtAddr           PhysAddr
        FileSiz            MemSiz              Flags  Align
--
  LOAD  0x0000000000000000 0x0000000000400000 0x0000000000400000
        0x00000000000006d4 0x00000000000006d4  R E    200000
  LOAD  0x0000000000000e10 0x0000000000600e10 0x0000000000600e10
        0x000000000000022c 0x0000000000000230  RW     200000
localhost ~ # objdump -d boom | grep retq | head -n1
  4003d5:       c3                      retq
localhost ~ # ./boom 0x4003d5
localhost ~ # readelf -s boom | grep ret_insns
  47: 0000000000601038     4 OBJECT  GLOBAL DEFAULT   24 ret_insns
localhost ~ # ./boom 0x601038
Segmentation fault (core dumped)
```

Compiler: -Wl,-z,relro

```
void main(int argc, char * argv[]) {
  unsigned char *ptr = (void*)strtoul(argv[1], NULL, 16);
  char cmd[80];  sprintf(cmd, "cat /proc/%d/maps", getpid());  system(cmd);
  printf("0x%02x 0x%02x 0x%02x 0x%02x 0x%02x\n",
         ptr[0], ptr[1], ptr[2], ptr[3], ptr[4]);
  printf("0x%02x 0x%02x 0x%02x 0x%02x 0x%02x\n",
         ptr[0], ptr[1], ptr[2], ptr[3], ptr[4]);
}
localhost ~ # gcc -Wl,-z,norelro plt.c -o lazy
localhost ~ # objdump lazy | grep '<printf@plt>:' -A1
0000000000400520 <printf@plt>:
  400520: ...  jmpq  *0x200662(%rip)  # 600b88 <_GLOBAL_OFFSET_TABLE_+0x30>
localhost ~ # ./lazy 600b88
00400000-00401000 r-xp 00000000 fc:00 159545   /tmp/lazy
00600000-00601000 rw-p 00000000 fc:00 159545   /tmp/lazy
...
0x26 0x05 0x40 0x00 0x00
0x40 0xb8 0x95 0x26 0x50
localhost ~ # gcc -Wl,-z,relro plt.c -o lazy
localhost ~ # objdump lazy | grep '<printf@plt>:' -A1
0000000000400520 <printf@plt>:
  400520: ...  jmpq  *0x200aca(%rip)  # 601030 <_GLOBAL_OFFSET_TABLE_+0x30>
localhost ~ # ./lazy 601030
00400000-00401000 r-xp 00000000 fc:00 159545   /tmp/lazy
00600000-00601000 r--p 00000000 fc:00 159545   /tmp/lazy
00601000-00602000 rw-p 00001000 fc:00 159545   /tmp/lazy
...
0x66 0x05 0x40 0x00 0x00
0x40 0xa8 0x9e 0xa0 0xee
```

Compiler: -Wl,-z,now

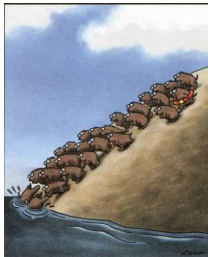
```
void main(int argc, char * argv[]) {
  unsigned char *ptr = (void*)strtoul(argv[1], NULL, 16);
  char cmd[80];  sprintf(cmd, "cat /proc/%d/maps", getpid());  system(cmd);
  printf("0x%02x 0x%02x 0x%02x 0x%02x 0x%02x\n",
         ptr[0], ptr[1], ptr[2], ptr[3], ptr[4]);
  printf("0x%02x 0x%02x 0x%02x 0x%02x 0x%02x\n",
         ptr[0], ptr[1], ptr[2], ptr[3], ptr[4]);
}
localhost ~ # gcc -Wl,-z,relro plt.c -o now
localhost ~ # objdump now | grep '<printf@plt>:' -A1
0000000000400520 <printf@plt>:
  400520: ...  jmpq  *0x200aca(%rip)  # 601030 <_GLOBAL_OFFSET_TABLE_+0x30>
localhost ~ # ./now 601030
00400000-00401000 r-xp 00000000 fc:00 159545   /tmp/now
00600000-00601000 r--p 00000000 fc:00 159545   /tmp/now
00601000-00602000 rw-p 00001000 fc:00 159545   /tmp/now
...
0x66 0x05 0x40 0x00 0x00
0x40 0xa8 0x9e 0xa0 0xee
localhost ~ # gcc -Wl,-z,relro -Wl,-z,now plt.c -o now
localhost ~ # objdump now | grep '<printf@plt>:' -A1
0000000000400520 <printf@plt>:
  400520: ...  jmpq  *0x200a72(%rip)  # 600fd8 <_GLOBAL_OFFSET_TABLE_+0x30>
localhost ~ # ./now 600fd8
00400000-00401000 r-xp 00000000 fc:00 159545   /tmp/now
00600000-00601000 r--p 00000000 fc:00 159545   /tmp/now
00601000-00602000 rw-p 00001000 fc:00 159545   /tmp/now
...
0x40 0x28 0x6c 0xd0 0x8e
0x40 0x28 0x6c 0xd0 0x8e
```

Userspace: ASLR

```
$ cat /proc/self/maps
```

```
00400000-0040b000 r-xp 00000000 fc:00 1069692          /bin/cat
0060a000-0060b000 r--p 0000a000 fc:00 1069692          /bin/cat
0060b000-0060c000 rw-p 0000b000 fc:00 1069692          /bin/cat

024c6000-024e7000 rw-p 00000000 00:00 0                [heap]

7f37e363e000-7f37e3d26000 r--p 00000000 fc:00 1181645  /usr/lib/locale/locale-archive
7f37e3d26000-7f37e3edb000 r-xp 00000000 fc:00 1052522  /lib/x86_64-linux-gnu/libc-2.15.so
7f37e3edb000-7f37e40da000 ---p 001b5000 fc:00 1052522  /lib/x86_64-linux-gnu/libc-2.15.so
7f37e40da000-7f37e40de000 r--p 001b4000 fc:00 1052522  /lib/x86_64-linux-gnu/libc-2.15.so
7f37e40de000-7f37e40e0000 rw-p 001b8000 fc:00 1052522  /lib/x86_64-linux-gnu/libc-2.15.so
7f37e40e0000-7f37e40e5000 rw-p 00000000 00:00 0
7f37e40e5000-7f37e4107000 r-xp 00000000 fc:00 1058112  /lib/x86_64-linux-gnu/ld-2.15.so
7f37e42cd000-7f37e42d0000 rw-p 00000000 00:00 0
7f37e4305000-7f37e4307000 rw-p 00000000 00:00 0
7f37e4307000-7f37e4308000 r--p 00022000 fc:00 1058112  /lib/x86_64-linux-gnu/ld-2.15.so
7f37e4308000-7f37e430a000 rw-p 00023000 fc:00 1058112  /lib/x86_64-linux-gnu/ld-2.15.so

7fff8191b000-7fff8193c000 rw-p 00000000 00:00 0        [stack]

7fff819ff000-7fff81a00000 r-xp 00000000 00:00 0        [vdso]

ffffffffff600000-ffffffffff601000 r-xp 0000000 00:00 0 [vsyscall]
```

- CONFIG_COMPAT_VDSO=n



Compiler: -fPIE -pie

- Generates relocatable main program code
  - most stuff already -fPIC for shared objects

```
localhost ~ # file $(which cat)
/bin/cat: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.9, ...
localhost ~ # cat /proc/self/maps | grep 'x.*cat'
7fe25240d000-7fe252418000 r-xp 00000000 08:03 33035  /bin/cat
localhost ~ # cat /proc/self/maps | grep 'x.*cat'
7f76c47a4000-7f76c47af000 r-xp 00000000 08:03 33035  /bin/cat
```

Userspace: glibc runtime checks

- Heap corruption
  - double-free, link validation, and on and on

```
localhost ~ # (cd malloc/; git grep -E 'malloc_printerr.*"|errstr = "')
hooks.c:  malloc_printerr (check_action, "malloc: top chunk is corrupt", t);
...
malloc.c:      errstr = "free(): invalid size";
malloc.c:           errstr = "free(): invalid next size (fast)";
malloc.c:           errstr = "double free or corruption (fasttop)";
malloc.c:       errstr = "invalid fastbin entry (free)";
...
```

- PTR_MANGLE for masking stored callbacks
  - XOR with global random value like stack canary
  - atexit, getXXbyYY, setjmp/longjmp

```
stored->callback = PTR_MANGLE(the_function);

...

func = PTR_UNMANGLE(stored->callback);
func();
```

Userspace: namespaces

- clone(func, stack, *flags*, arg...)
  - CLONE_NEWIPC
    - can't hear you
  - CLONE_NEWNET
    - don't forget to bring up lo interface
  - CLONE_NEWNS
    - stronger than chroot
  - CLONE_NEWPID
    - always enjoy getpid() returning 1
  - CLONE_NEWUTS
    - your own private hostname

Userspace: ptrace restrictions

```
localhost ~ # su - -c /bin/bash chronos
chronos@localhost ~ # sleep 60 &
[1] 10027
chronos@localhost ~ # gdb
...
(gdb) attach 10027
Attaching to process 10027
ptrace: Operation not permitted.
(gdb) quit

chronos@localhost ~ # gdb $(which sleep)
...
(gdb) run 1
...
Program exited normally.
chronos@localhost ~ # exit
localhost ~ # dmesg | grep ptrace
[9371.184349] ptrace of non-child pid 10027 was attempted by: gdb (pid 10037)
```

- ... and declared ptracers
  - prctl(PR_SET_PTRACER, tracer_pid, ...);

Userspace: link restrictions

```
localhost ~ # umask 066
localhost ~ # echo b-sides >/home/sekrit
localhost ~ # ls -ld /tmp
drwxrwxrwt 10 root root 960 Nov  8 17:21 /tmp
localhost ~ # ln -s /etc/passwd /tmp/good
localhost ~ # ls -l /tmp/good
lrwxrwxrwx 1 root root 11 Nov  8 17:33 /tmp/good -> /etc/passwd
localhost ~ # head -n1 /tmp/good
root:x:0:0:root:/root:/bin/bash
localhost ~ # su - -c /bin/bash chronos

chronos@localhost ~ # ln /home/sekrit /home/chronos/evil
ln: creating hard link `/home/chronos/evil' => `/home/sekrit': Operation not permitted
chronos@localhost ~ # head -n1 /tmp/good
root:x:0:0:root:/root:/bin/bash
chronos@localhost ~ # ln -s /etc/shadow /tmp/evil
chronos@localhost ~ # ls -l /tmp/evil
lrwxrwxrwx 1 chronos chronos 11 Nov  8 17:31 /tmp/evil -> /etc/shadow
chronos@localhost ~ # head -n1 /tmp/evil
head: cannot open `/tmp/evil' for reading: Permission denied
chronos@localhost ~ # exit

localhost ~ # head -n1 /tmp/evil
head: cannot open `/tmp/evil' for reading: Permission denied
```

Kernel: stack-protector

- CONFIG_CC_STACKPROTECTOR=y

```
localhost ~ # readelf -s /lib/modules/*/kernel/fs/fat/vfat.ko | grep _chk
    42: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT  UND __stack_chk_fail

localhost ~ # grep -A4 ' __stack_chk_fail' kernel/panic.c
void __stack_chk_fail(void)
{
        panic("stack-protector: Kernel stack is corrupted in: %p\n",
                __builtin_return_address(0));
}
```

Kernel: memory access restriction

- CONFIG_DEFAULT_MMAP_MIN_ADDR=64k

```
struct function_table {
     void(*something)(int arg),
     ...
};

struct function_table foo;
memset(&foo, 0, sizeof(foo));
...
foo.something(9001);
```

```
int prot=PROT_WRITE | PROT_EXEC;
int flags=MAP_PRIVATE | MAP_FIXED;
void *addr=0x0;
unsigned int *var;

var=mmap(addr, getpagesize(),
         prot, flags, -1, 0);
*var = 0xc3c3c3c3;
```

- CONFIG_STRICT_DEVMEM=y
  - /dev/mem maps to physical memory, which is more than just system RAM. Go see all the fun stuff in /proc/iomem.
- CONFIG_DEVKMEM=n
  - just removing /dev/kmem isn't going to cut it

Kernel: RO/NX

- CONFIG_DEBUG_RODATA=y
- CONFIG_DEBUG_SET_RONX=y

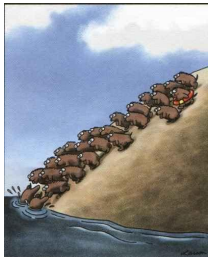
- use CONFIG_X86_PTDUMP to visualize

```
localhost ~ # cat /sys/kernel/debug/kernel_page_tables
---[ User Space ]---
0x0000000000000000-0xffff800000000000 16777088T                          pgd
---[ Kernel Space ]---
0xffff800000000000-0xffff880000000000        8T                          pgd
---[ Low Kernel Mapping ]---
0xffff880000000000-0xffff880000097000      604K     RW             GLB x  pte
0xffff880000097000-0xffff880000098000        4K     ro             GLB NX pte
...
---[ vmalloc() Area ]---
0xffffc90000000000-0xffffc90000001000        4K     RW     PCD     GLB NX pte
0xffffc90000001000-0xffffc90000002000        4K                            pte
...
```

Kernel: kptr_restrict

- /proc/sys/kernel/kptr_restrict
  - 0 for everyone
  - 1 for CAP_SYSLOG
  - 2 for nobody
- Marked in kernel source with format string flag and conversion specifier: %pK

```
localhost ~ # grep %pK kernel/kallsyms.c
                seq_printf(m, "%pK %c %s\n", (void *)iter->value,

localhost ~ # grep do_signal_stop /proc/kallsyms
ffffffff81039c29 t do_signal_stop
localhost ~ # su - -c /bin/bash chronos
chronos@localhost ~ # grep do_signal_stop /proc/kallsyms
0000000000000000 t do_signal_stop
```



Doing what others will be doing

Doing what others will be doing

- Use -fstack-protector-strong
  - all arrays, address on RHS or passed to func
- Confinement with seccomp
  - can't attack the kernel if you can't talk to it
- Verified boot
  - imposes a bright line between uid 0 and ring 0
  - no hibernation, kexec
- Evaluating D-Bus exposure
  - likely need explicit confinement

-fstack-protector-strong

- All arrays of any type, even in unions/structs
- Stack address used in RHS of assignment
- Stack address passed to function call

http://gcc.gnu.org/ml/gcc-patches/2012-10/msg00208.html

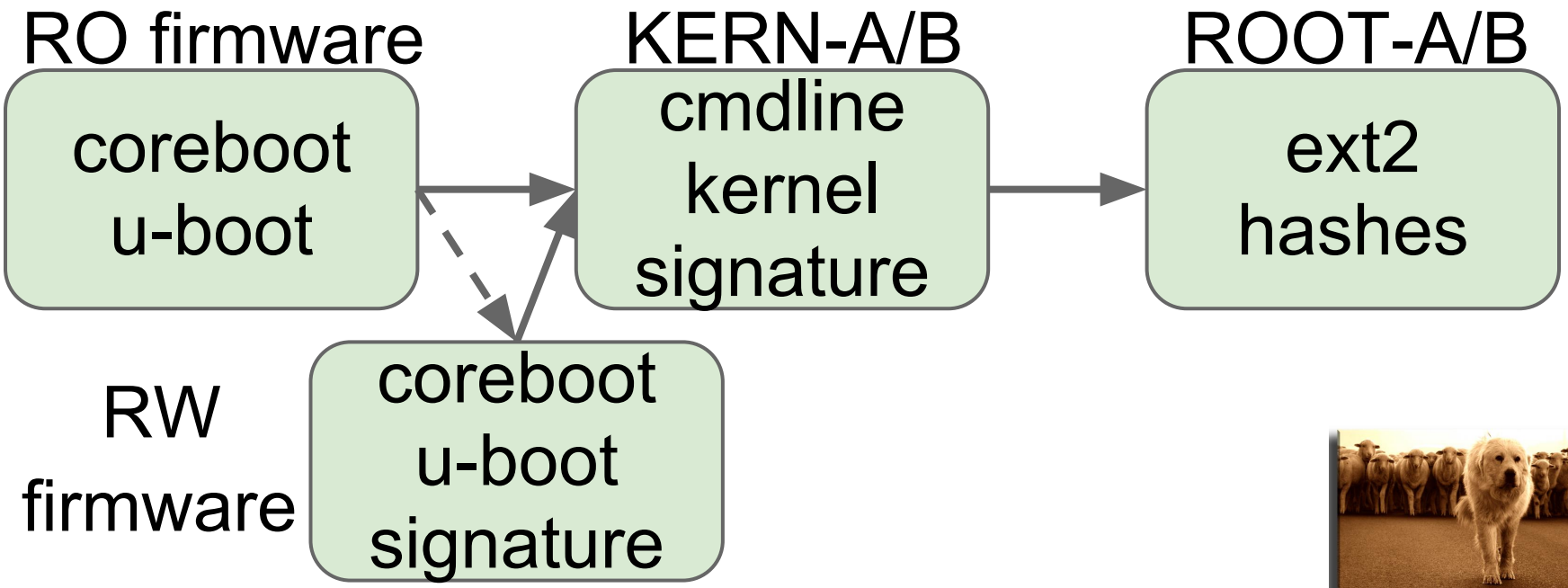
Confinement with seccomp

- Deeper sandboxing
  - DAC: generally user-controlled (e.g. private home directory, chroot)
  - MAC: generally sysadmin-controlled (e.g. AppArmor, SELinux)
  - syscall filtering: generally programmer-controlled
- BPF filters for examining syscalls
  - kill, trap, errno
- Already filtering:
  - Chrome
  - various daemons

http://www.outflux.net/teach-seccomp/

Verified boot

- Bright line between uid 0 and ring 0
- CONFIG_KEXEC=n
- CONFIG_HIBERNATION=n



Evaluating D-Bus exposure

- Daemons listening with various APIs
- Usually with root privileges
- Not already running? Can auto-launch from /usr/share/dbus-1/system-services/

```
localhost ~ # dbus-send --print-reply --system --dest=org.freedesktop.DBus
/org/freedesktop/DBus org.freedesktop.DBus.ListNames | grep string | wc -l
40
```

Doing what others haven't done

Doing what others haven't done

- Coreboot as firmware
  - developer mode, no PC-BIOS, no UEFI
- Privacy-safe remote attestation
  - content licensing
- Read-only root filesystem and dm-verity
  - general distros don't want to face this
- Further glibc hardening
  - FILE structure pointer mangling
- Kernel module loading restrictions
  - very useful with a read-only filesystem
- Extracting bug fixes from grsecurity
  - spender loves fixing bugs

Coreboot as firmware

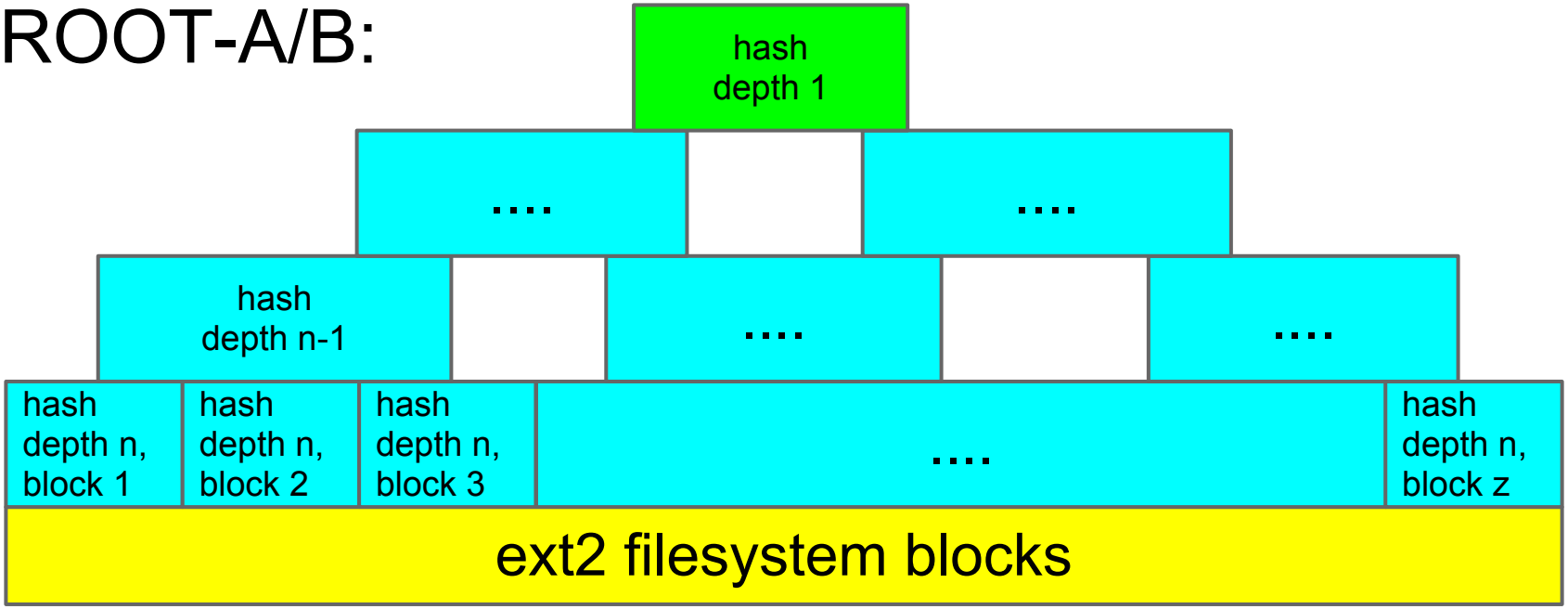
- No PC-BIOS, no UEFI
- RO (hardware write-protected)
- RW with signature
- Developer mode
  - requires physical presence (switch or keyboard lines)

Privacy-safe remote attestation

- Content owners will only license playback when there is ample reason to believe they won't lose control of content
- Enterprises can trust device for VPN access
- Remotely attest:
  - a Chromebook is in use (TPM factory key)
  - the Chromebook is in verified mode (RO firmware)
  - signed random identifier *for each third party service* which hinders third party collusion to identify users

Read-only root fs and dm-verity

ROOT-A/B:

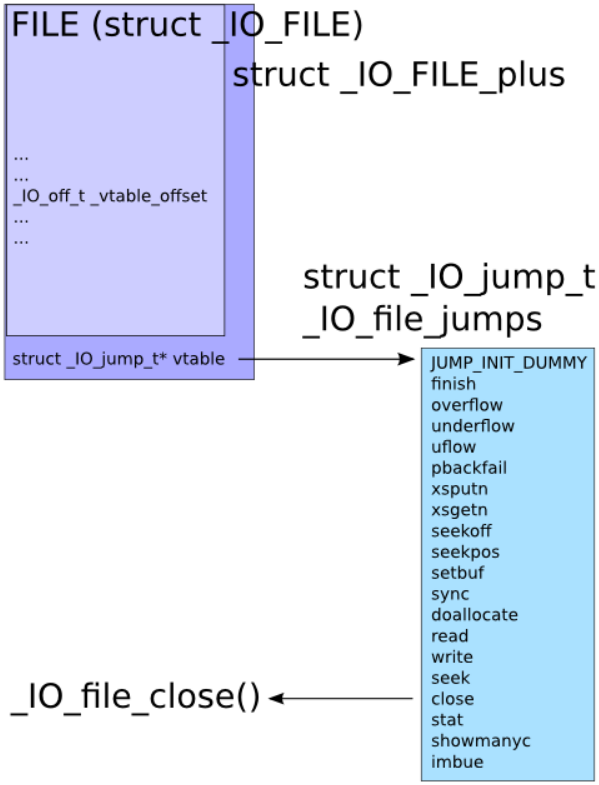



```
localhost ~ # cat /proc/cmdline
... root=PARTUUID=200421c7-.../PARTNROFF=1 dm="vroot none ro,0 2097152 verity
... hashstart=2097152 alg=sha1
root_hexdigest=3f127335bb695420521f6ef4b9b1a3a039207c3c ..." ...
```

Further glibc hardening

http://www.outflux.net/blog/archives/2011/12/22/abusing-the-file-structure/

- Use PTR_MANGLE on FILE structure




```
#include <stdio.h>
void main(void) {
    char cmd[80];
    sprintf(cmd, "cat /proc/%d/maps", getpid());
    system(cmd);
    FILE *fp = fopen("/dev/null", "w");
    printf("%p\n", *(void**)(fp + 1));
    *(unsigned long*)(fp + 1) = 0;
    fclose(fp);
}
localhost ~ # gdb -ex run ./boom
...
7fb87381e000-7fb873822000 r--p ... /lib/libc-2.15.so
...
0x7fb873821660
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7a9538a in _IO_new_file_close_it
(fp=0x602010) at fileops.c:177
(gdb)
```

Kernel module loading restrictions

- Current interface:

```
int init_module(void *module_image,
                unsigned long len,
                const char *param_values);

char evil[] = "pwn yer kernelz";
init_module(evil, sizeof(evil), "");
```

- New interface:

```
int finit_module(int fd,
                 const char *param_values,
                 int flags);

int fd = open("/lib/modules/.../vfat.ko", O_RDONLY);
finit_module(fd, "", 0);
```

- Very useful on a read-only filesystem

Extracting bug fixes from grsecurity

- spender loves fixing bugs

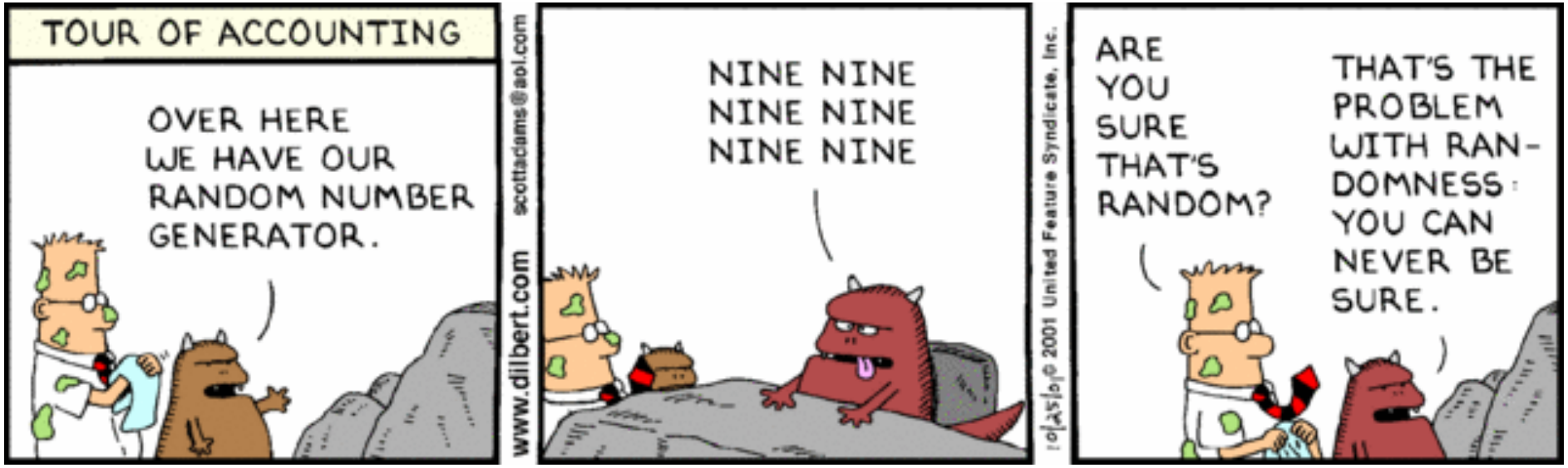
```
localhost ~ # less changelog-test.txt    [1]
...
commit 76f58d8cf81aae8bb61bc88a60b4d924dbb46839
Author: Brad Spengler <spender@grsecurity.net>
Date:   Wed Sep 19 18:46:35 2012 -0400

    Fix 3.x uname emulation infoleak

 kernel/sys.c |   14 ++++++++-------
 1 files changed, 7 insertions(+), 7 deletions(-)
...
```

- I like ~~getting abused by~~ sending them upstream

http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=commitdiff;h=2702b1526c7278c4d65d78de209a465d4de2885e CVE-2012-0957

1. ^ changelog for the test series of patches. grsecurity.net/changelog-test.txt. Retrieved 2012-09-19.

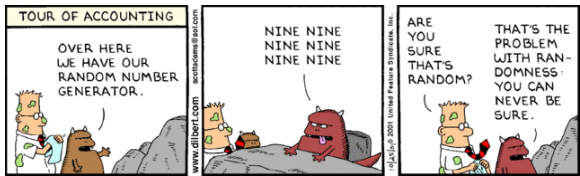
Testing ourselves



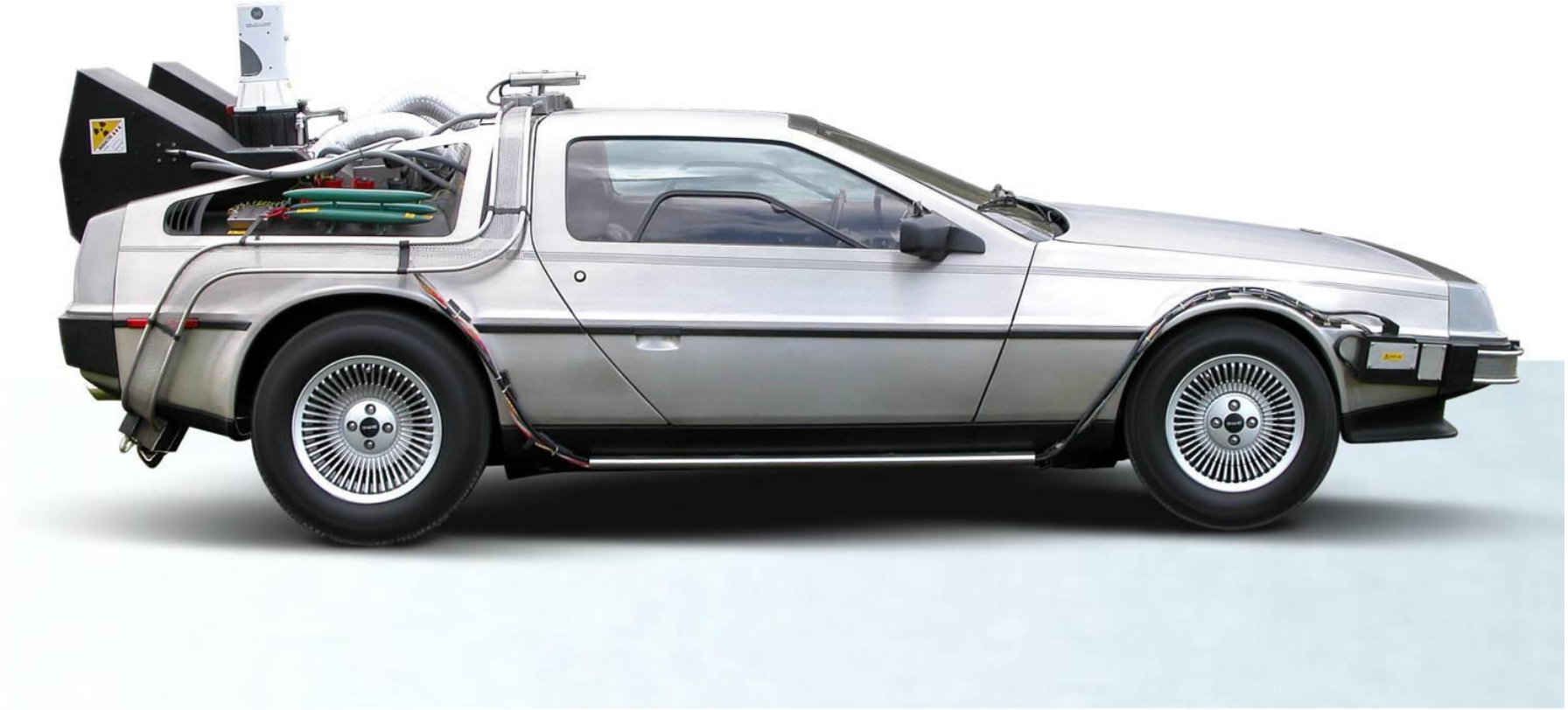
Testing ourselves

- Use autotest with test images
  - security test in with the rest of Chrome OS
- Use external tests with signed images
  - what couldn't be tested on a test image?

- Quantification of attack surface
  - "more ports, suid binaries, IPC, symlinks, Chrome extensions?"
- Testing hardening for regressions
  - configuration: "are we built with ASLR?"
  - behavior: "is ASLR working?"



Seeing the future

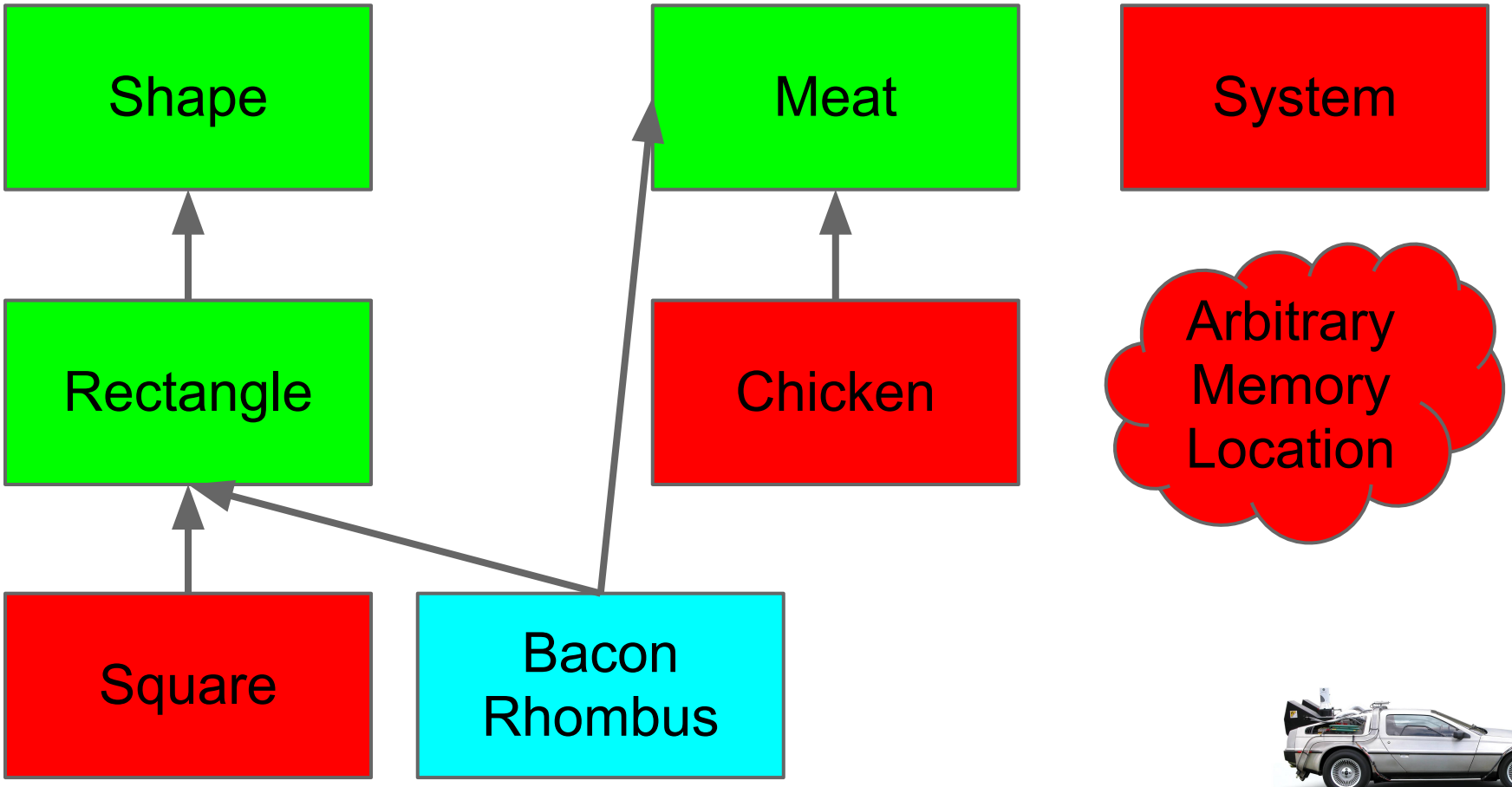


Seeing the future

- More userspace hardening
  - vtable ancestry checking
- Mount W^X
  - never let userspace remount exec
- Improved encryption
  - dm-thin for sharing
- Aggressive kernel hardening
  - kernel ASLR
  - looking to PaX and grsecurity

More userspace hardening

- C++ object vtable ancestry checking



Mount W^X

- Never let userspace change dangerous mount options via "mount -o remount,...":
  - noexec
  - nosuid
  - nodev

Improved encryption

- Replace eCryptfs
  - performance issues
  - corner-case behavior
- Use dm-thin for shared provisioning
  - still young

Aggressive kernel hardening

- Set /proc/sys/kernel/dmesg_restrict=1
  - so much stuff wants to read dmesg
- Kernel ASLR
  - so many corner cases
- Looking to PaX and grsecurity
  - more RO (e.g. function pointer constification)
  - leak protections (e.g. stack clearing gcc plugin)
  - emulated SMEP/SMAP (UDEREF)
  - anti-ROP (KERNEXEC)

Resources

- http://code.google.com/p/chromium-os/wiki/SystemHardeningFeatures
- https://wiki.ubuntu.com/ToolChain/CompilerFlags

- chromeos-security@google.com
- keescook@{google.com,chromium.org}
- kees@outflux.net