Linux Kernel Exploitation
Earning Its Pwnie a Vuln at a Time

Jon Oberheide

CTO, Scio Security

An Ode to My Dear Eyjafjallajokull

BEWARE THE CYBER POMPEII

http://jon.oberheide.org/files/cyber-pompeii.txt

The Jono

- Jon Oberheide
  - BS, MS, and PhD (ABD) from U of M
  - CTO of Scio Security
- What we're talking about today
  - Linux kernel security!

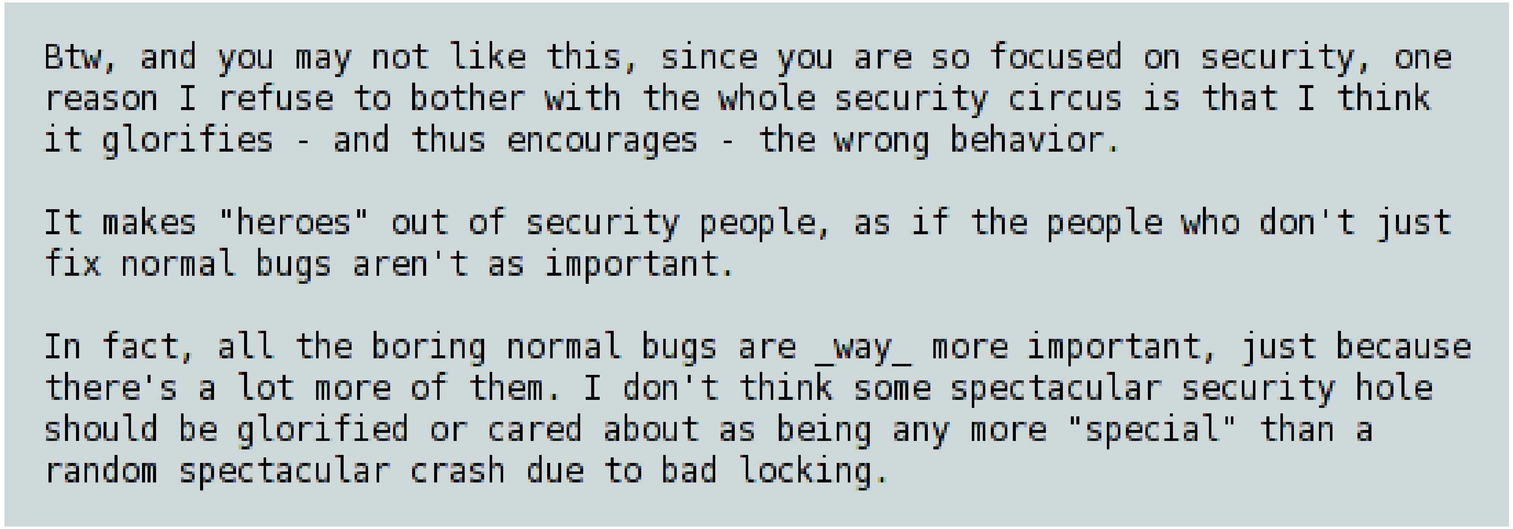
Agenda

- **Why the Linux Kernel?**
- A History of Vulns
- Vulnerability Classes
- Wrap-up

Why the Linux Kernel?

- Administrators:
  - Know your exposure!
- Researchers:
  - Relatively soft target
  - Lots of interesting vulndev opportunities
- Security mechanisms not deployed
  - Available in external patches, not in mainline
  - Execution overhead can be nontrivial

Linus Wins a Pwnie!



```
Btw, and you may not like this, since you are so focused on security, one
reason I refuse to bother with the whole security circus is that I think
it glorifies - and thus encourages - the wrong behavior.

It makes "heroes" out of security people, as if the people who don't just
fix normal bugs aren't as important.

In fact, all the boring normal bugs are _way_ more important, just because
there's a lot more of them. I don't think some spectacular security hole
should be glorified or cared about as being any more "special" than a
random spectacular crash due to bad locking.
```

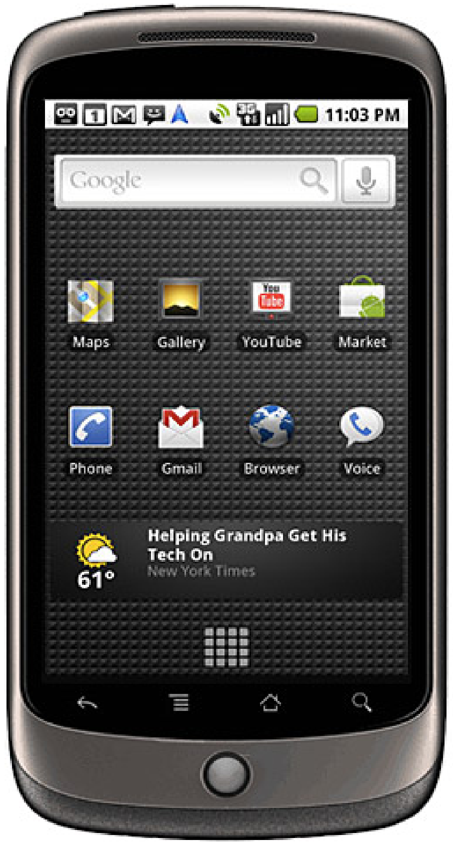
- Good: distro in charge of security!
- Bad: distro in charge of security!
- Eugene++

Bug Misclassification

- Many have non-obvious security impact
  - Often “silently” fixed
  - Intentional or not, same impact
- Miscommunication between devs/distros
- Attackers may classify better!
  - Unfortunate side-effect of open development

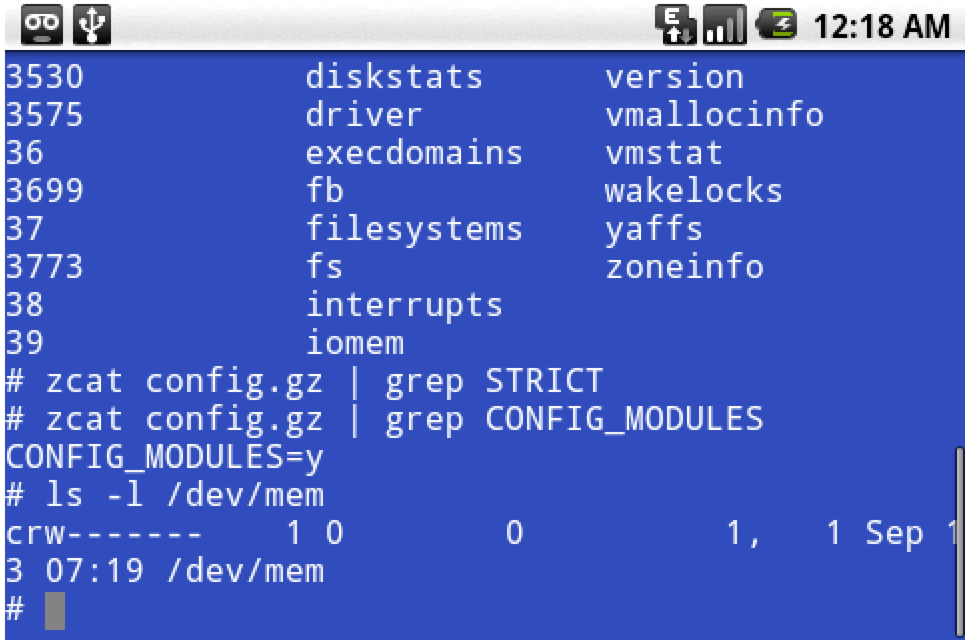
Embedded Devices: Mobile!

- My mobile phone!
  - Android, Bada, MeeGo, WebOS
  - 1990s exploitability
  - Lack of user patching
  - Numerous untrusted users (aka apps)
- Third-party “sleeper” apps
  - Legitimate looking app/game
  - Get solid user base of installs
  - Wait for privesc, deliver payload, rootkit!
  - Easy to win race against provider



Android Privilege Escalation

- Zinx port of spender's wunderbar_emporium [1,2]
  - Didn't have MMAP_MIN_ADDR
  - Map get_root() object code at 0x0
  - Trigger sock_sendpage() NULL func ptr deref
  - Root archived!
- Rootkit loading
  - CONFIG_MODULES=y
  - /dev/mem unrestricted



```
3530           diskstats      version
3575           driver         vmallocinfo
36             execdomains    vmstat
3699           fb             wakelocks
37             filesystems    yaffs
3773           fs             zoneinfo
38             interrupts
39             iomem
# zcat config.gz | grep STRICT
# zcat config.gz | grep CONFIG_MODULES
CONFIG_MODULES=y
# ls -l /dev/mem
crw-------    1 0        0          1,    1 Sep 1
3 07:19 /dev/mem
# 
```

Embedded Devices: TVs!

- My LG TV!
  - MIPS box
  - Flash malicious firmware via USB
  - Pop root shell via serial
  - Now with built-in ethernet/wifi........

```
Linux version 2.6.26 (gunhoon@swfarm-l1) (gcc version 3.4.3 (MontaVista 3.4.3-25.0.70.0501961 2005-12-18)) #108 PREEMPT Wed Nov 4 09:22:14 KST 2009
```

Kernel vs. Userspace

Traditional memory corruption vulns are boring!

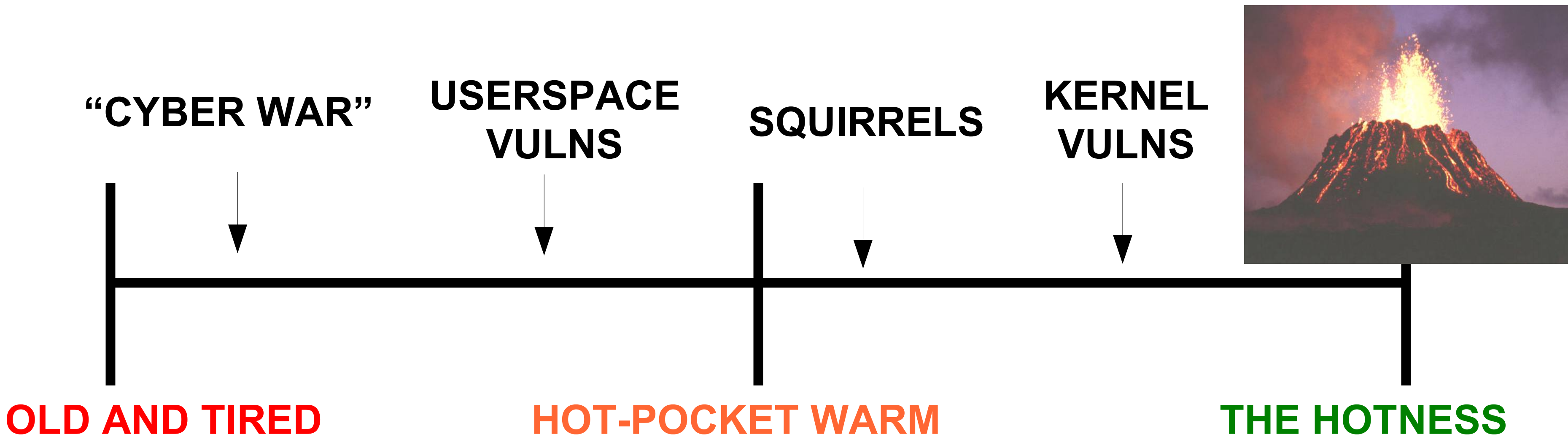
- Userspace increasingly hardened
  - NX + ASLR + PIE + RELRO + canaries = hard.
- Non-traditional vulns more interesting
  - A couple in userspace (eg. udev vuln [3,4])
  - A whole lot more in kernel space
  - Require semantic knowledge of a particular subsystem
  - Pros: interesting vulndev; Cons: non-reusable patterns

> 60% of local privilege escalations are not traditional stack/heap memory corruption

Kernel vs. Userspace

THE INFOSEC THERMOMETER!!!

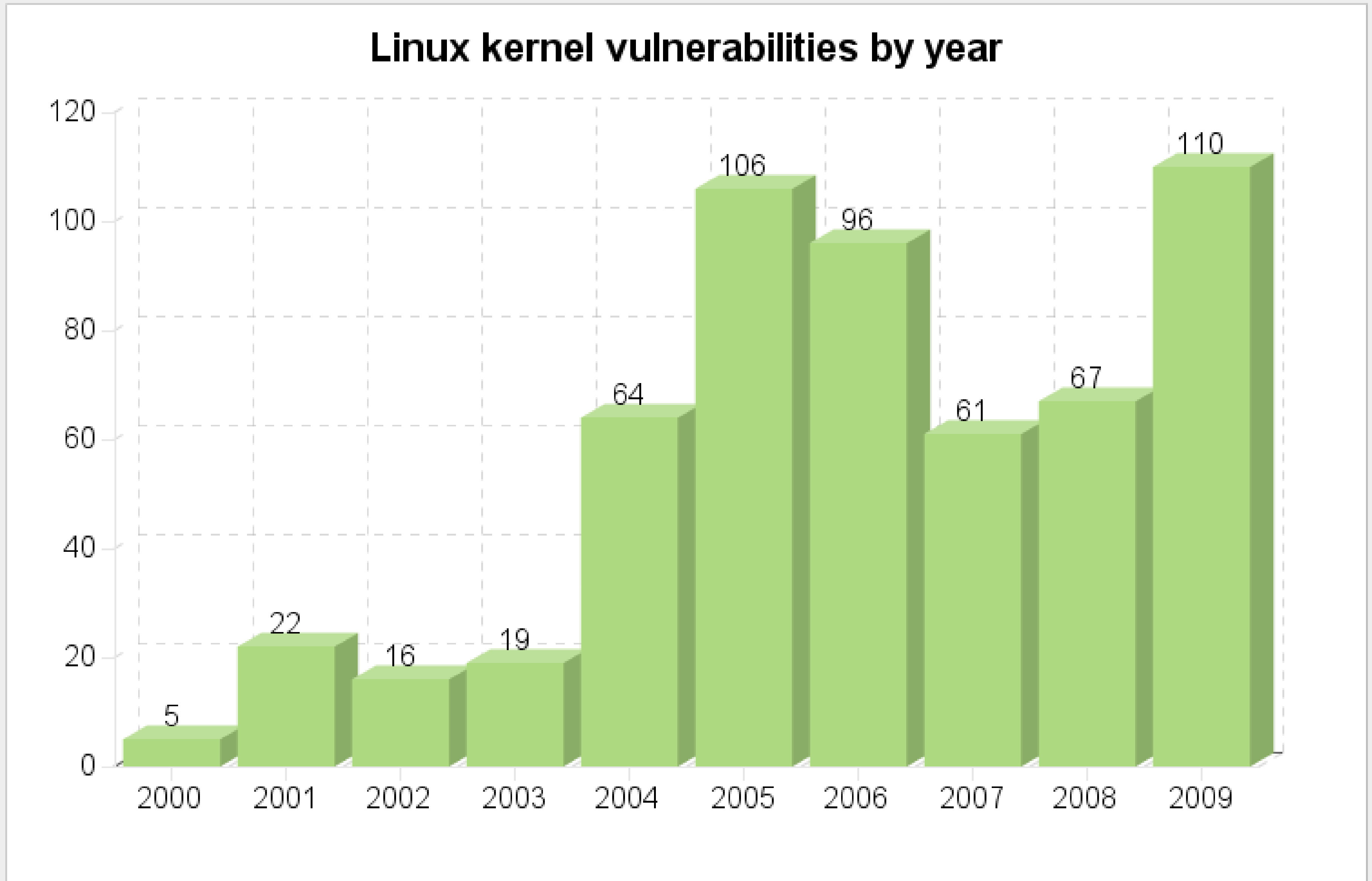
Advanced
Pompeii
Threats



Agenda

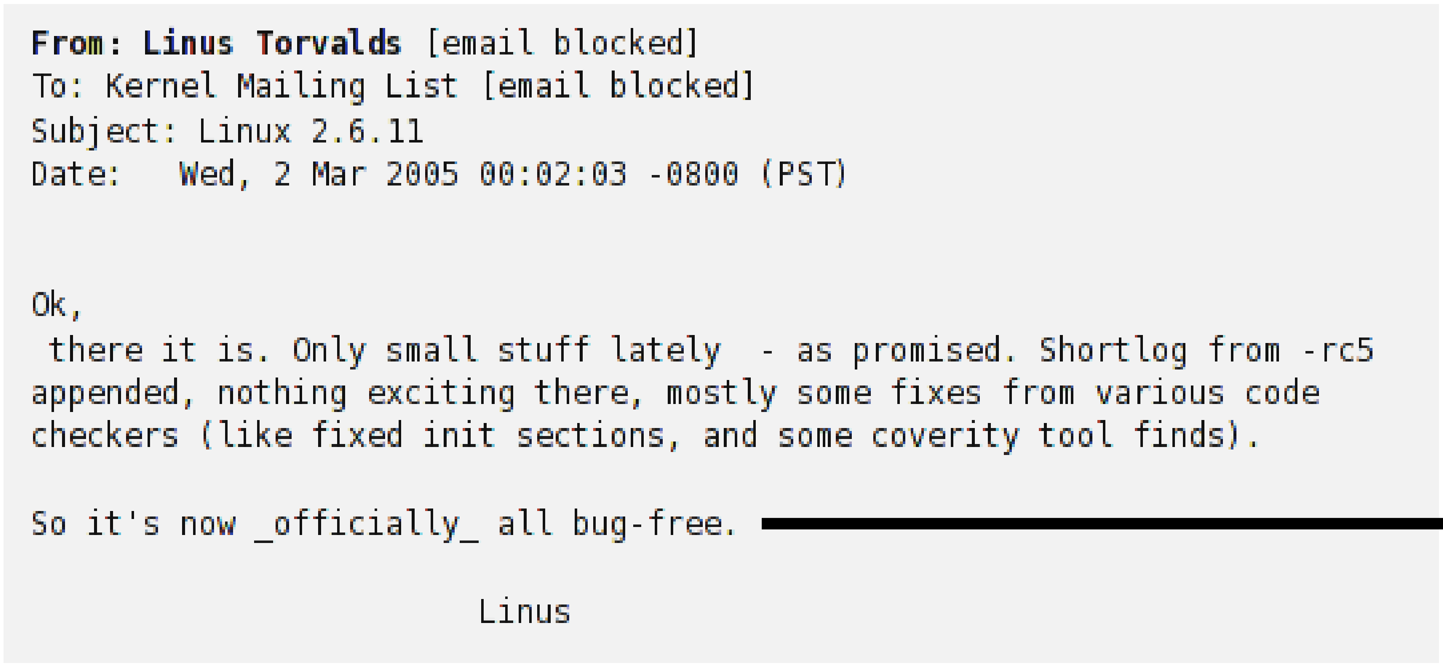
- Why the Linux Kernel?
- **A History of Vulns**
- Vulnerability Classes
- Wrap-up

Vulnerabilities By Year

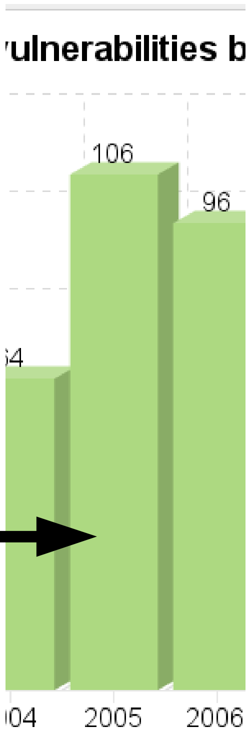


2005?!?

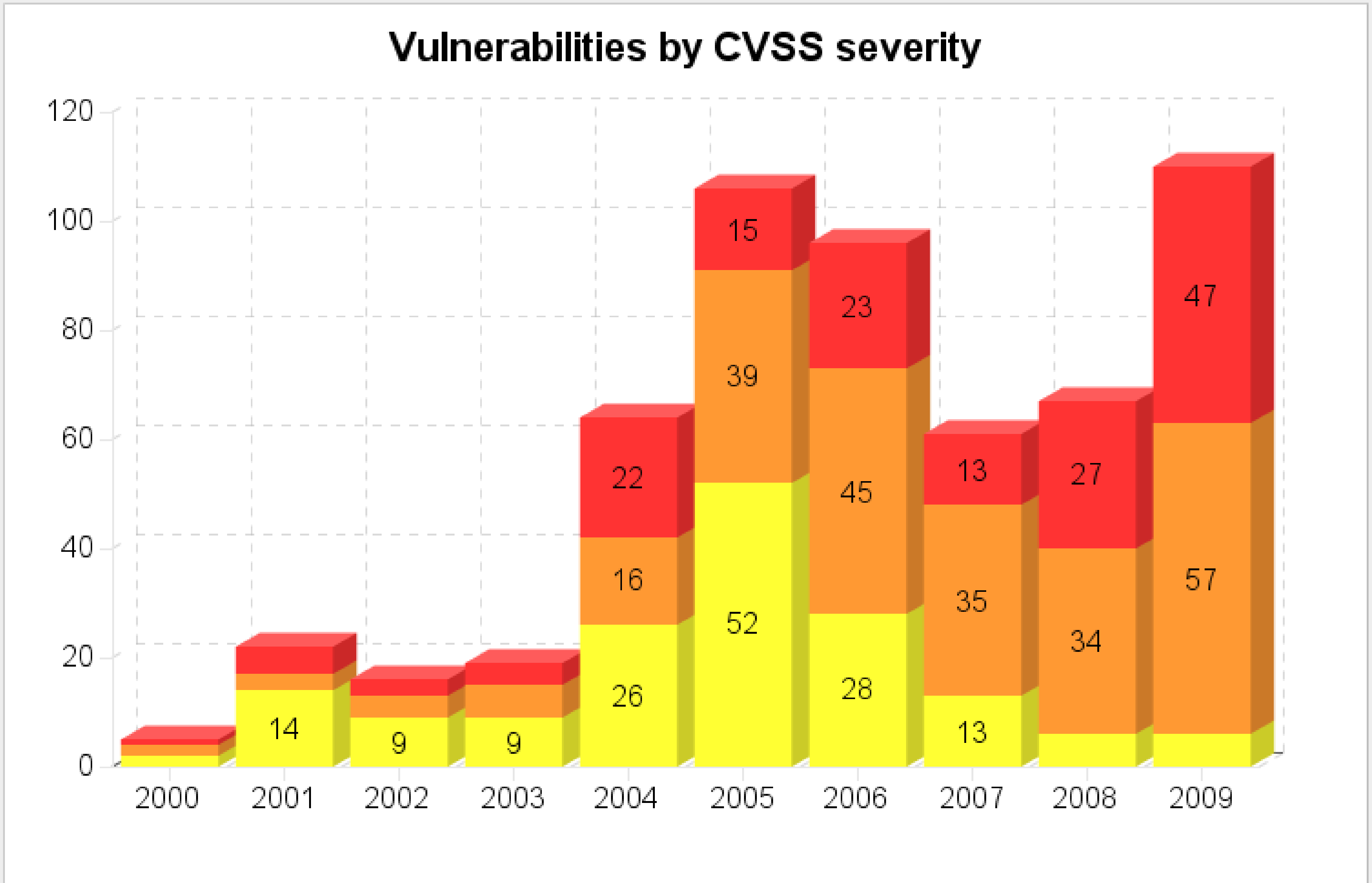
- 2.6.11 – first release of 2005
  - Start of the 2.6.x.y stable versioning scheme



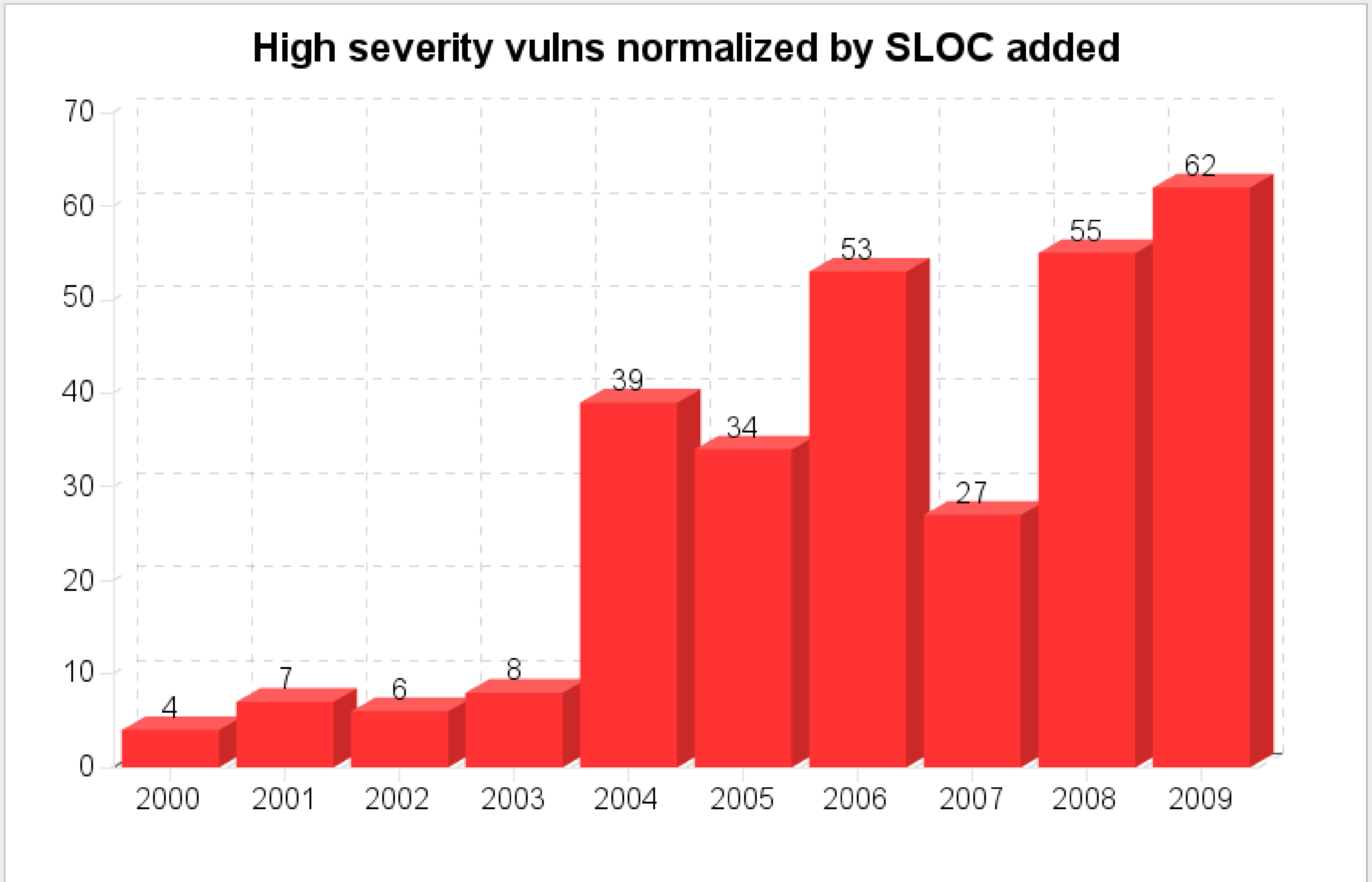
```
From: Linus Torvalds [email blocked]
To: Kernel Mailing List [email blocked]
Subject: Linux 2.6.11
Date:   Wed, 2 Mar 2005 00:02:03 -0800 (PST)


Ok,
 there it is. Only small stuff lately  - as promised. Shortlog from -rc5
appended, nothing exciting there, mostly some fixes from various code
checkers (like fixed init sections, and some coverity tool finds).

So it's now _officially_ all bug-free.

                        Linus
```



Vulnerability Severity

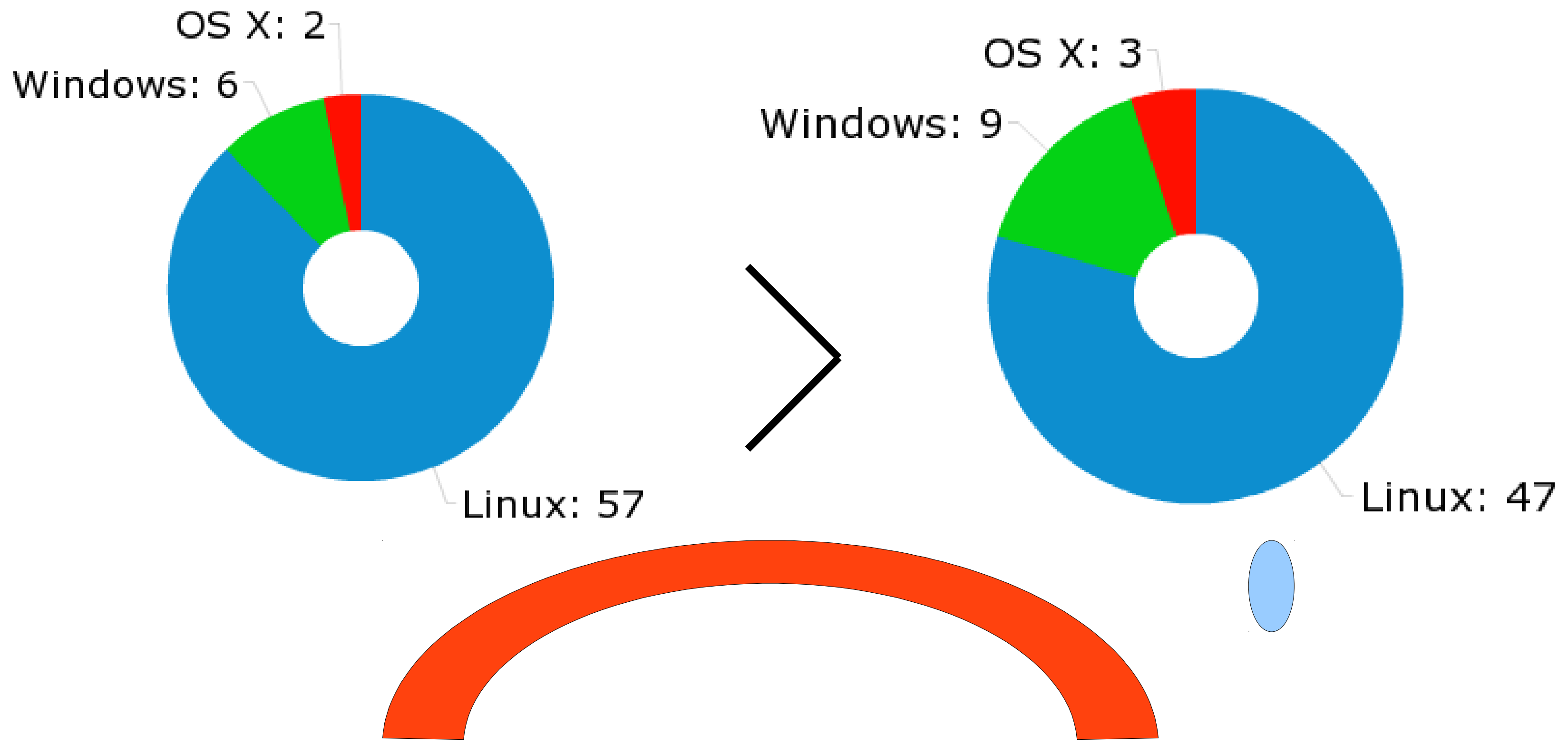


Vulnerabilities by SLOC



Other OS Kernel Vulns

2009 Medium CVSS Severity **2009 High CVSS Severity**



The Linux kernel averaged > 2 vulns/week for all of 2009.

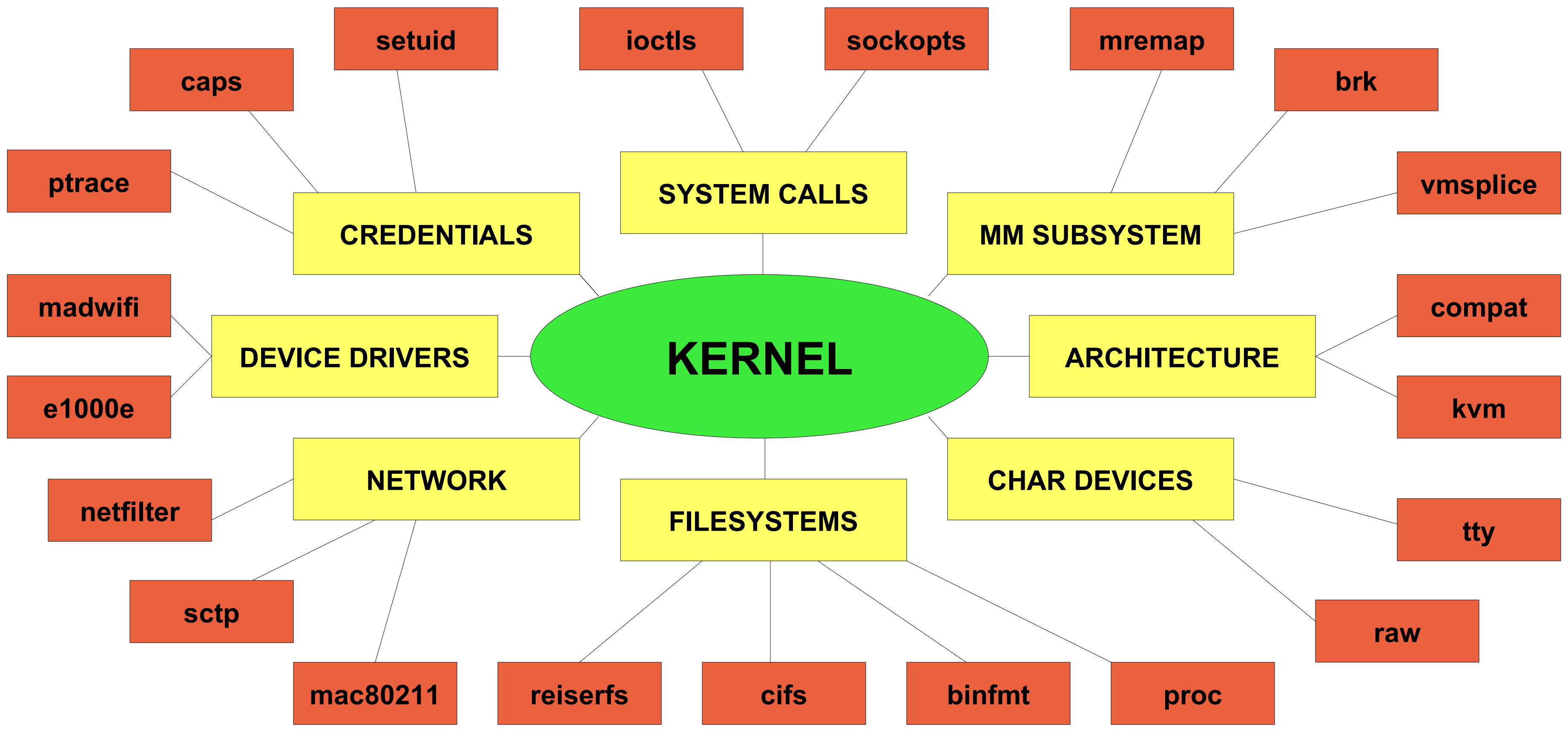
Agenda

- Why the Linux Kernel?
- A History of Vulns
- **Vulnerability Classes**
- Wrap-up

Exploit Vectors

- Remote
  - Fairly rare
  - sgrakkyu's sctp_houdini [5], Julien's madwifi [6]
  - Interrupt context issues [7]
- Local
  - Quite common
  - Mostly in form of privilege escalation
  - Kernel has a fairly large attack surface

Kernel Attack Surface



Contrary to popular belief, most vulns are *<u>not</u>* in device drivers.

Let's Look at a Recent Example

- Vulnerability in ReiserFS filesystem [8]
- .reiserfs_priv fails to enforce perms
  - A virtual path, you won't see this is “ls -la /”
  - Internal filesystem usage
- Including extended attribute (xattr) storage
  - We can write xattrs for arbitrary files!

ReiserFS Exploit

- Ok, arbitrary xattrs, how to escalate?
  - One xattr use: POSIX file capabilities
  - Finer-grained privs than setuid bit (eg. CAP_NET_RAW)
- CAP_SETUID sounds good!
  - Let's apply it to our own shell!
- Ok, how to get inode/object id of shell?
  - Compile shell, getdents(), set dummy xattr, getdents()
  - Write out CAP_SETUID xattr, exec the shell, root!
- Demo against fully patched Ubuntu

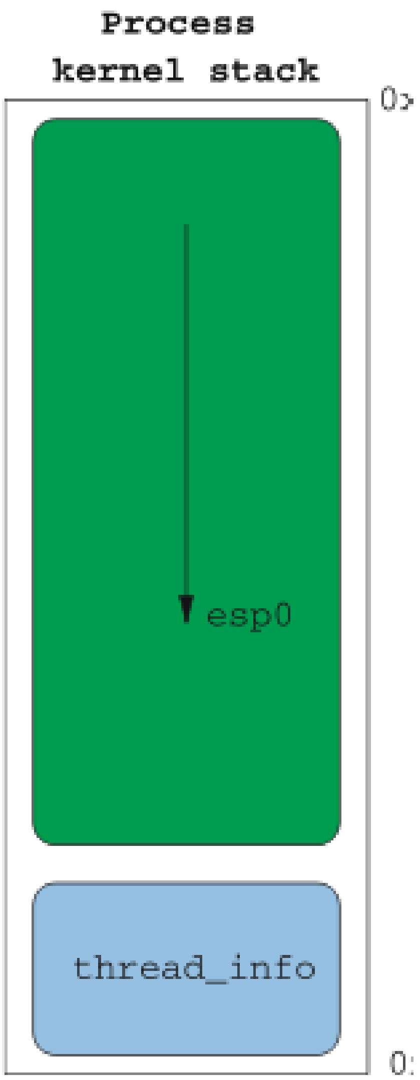
http://jon.oberheide.org/files/team-edward.py

What's Interesting Here?

- No memory corruption here!
  - Just a bit of fs/xattr/caps knowledge
  - Not a repeatable bug class / exploit pattern
- Representative of many Linux kernel vulns
  - Lots of one-off logic bugs like this one
  - But we also have the “standard” classes, too

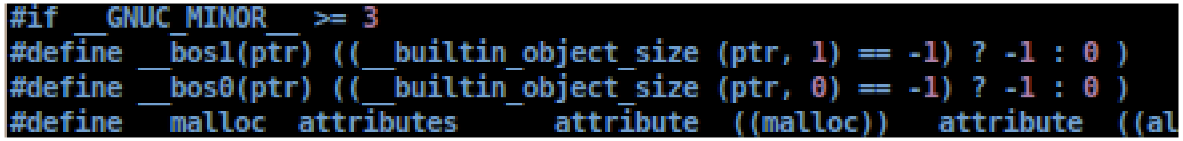
Kernel Stack Smashing

- If you can do userland, you can do kernel
- We have existing protection mechanisms
  - Stack canaries, bounds checking, etc
  - But, how effective in kernel space?
  - Not so great... [9,10]



FORTIFY_SOURCE Coverage

- FORTIFY_SOURCE
  - Compile-time bounds checking on potentially unsafe stack operations (strcpy/memcpy/etc)
- gcc __builtin_object_size [11]



```
#if __GNUC_MINOR__ >= 3
#define __bos1(ptr) ((__builtin_object_size (ptr, 1) == -1) ? -1 : 0 )
#define __bos0(ptr) ((__builtin_object_size (ptr, 0) == -1) ? -1 : 0 )
#define __malloc_attributes __attribute__ ((malloc)) __attribute__ ((al
```

```
$ make allyesconfig && make
$ cat success.txt | wc -l
5725
$ cat fail-o-rama.txt | wc -l
11638
```

- Total: 5725 +11638 = 17363
- Coverage: 5725 / 17363 = **32.9%**

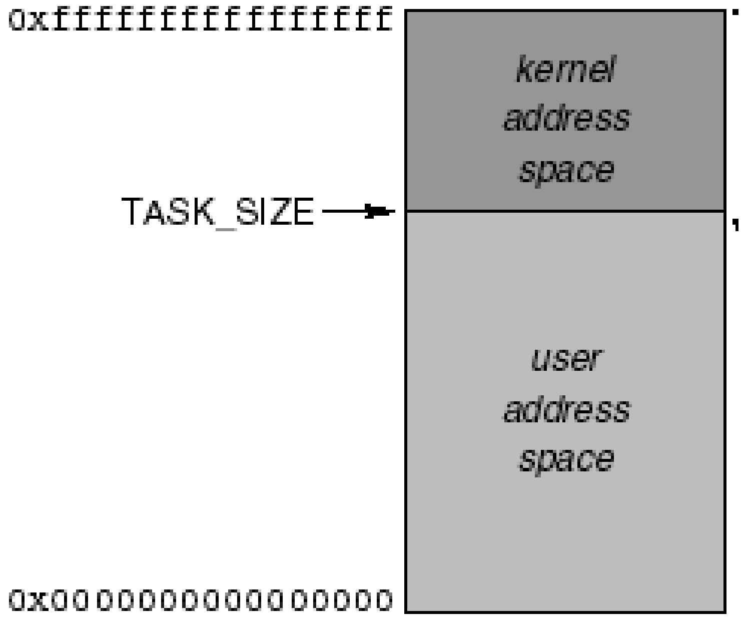
GCC's FORTIFY_SOURCE covers < 1/3 of bounds checking cases.

Kernel Heap Overflows

- SLAB/SLUB/SLOB allocators
  - SLUB is default on current kernels
  - Provides the kmalloc() family
- Must reads:
  - tioctl/sctp_houdini.c [12,5]
  - Larry H's Phrack 66 article [13]
- KERNHEAP Heap Protection [14]
  - Metadata protection, safe unlinking, etc

Userland Pointer Dereference

- Often misclassified as NULL ptr derefs
  - NULL is just a specific case (address 0x0)
  - Happens to be more common due to programming mistakes
- Shared VM architectures
  - Userspace and kernel in single VM space
  - Unsafe deference of userspace pointers by kernel code
  - Usually FP deref or data control converted later

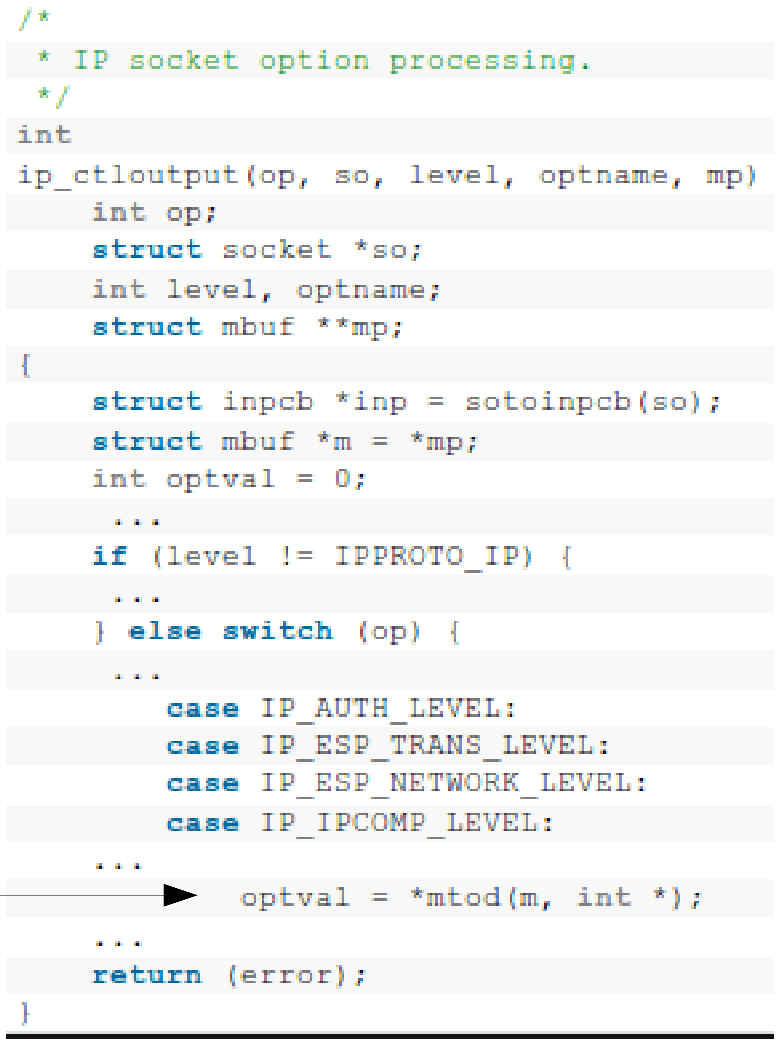


Traditional Exploitation Pattern

- Assuming NULL deref of FP
- Set up your payload
  - mmap(0, …)
  - memcpy(0, shellcode, sizeof(shellcode));
- Trigger the NULL deref
  - sendfile(fd1, fd2, NULL, 4096);
- See enlightenment framework [15]
  - Just fill in your prepare/trigger functions and boom!

An Example From...OpenBSD?!?

- Ok, it's not Linux, I'm cheating!
  - But it's multi-stage, kind of cute
- getsockopt() [16]
  - NULL dereference
  - Controlled write of 0x1



```
/*
 * IP socket option processing.
 */
int
ip_ctloutput(op, so, level, optname, mp)
	int op;
	struct socket *so;
	int level, optname;
	struct mbuf **mp;
{
	struct inpcb *inp = sotoinpcb(so);
	struct mbuf *m = *mp;
	int optval = 0;
	 ...
	if (level != IPPROTO_IP) {
	 ...
	} else switch (op) {
	 ...
		case IP_AUTH_LEVEL:
		case IP_ESP_TRANS_LEVEL:
		case IP_ESP_NETWORK_LEVEL:
		case IP_IPCOMP_LEVEL:
	...
			optval = *mtod(m, int *);
	...
	return (error);
}
```

Exploit Procedure

USERSPACE **KERNEL**

1. Call mmap to map zero page

2. Set up fake mbuf at 0x0
mbuf->data should point to
address of a syscall table entry

3. Call getsockopt to trigger vuln

4. Kernel accesses fake mbuf
located at 0x0, writes value 0x1
to address in mbuf->data,
overwriting syscall entry

5. Set up root-getting shellcode
at address 0x1

6. Call syscall that we overwrote

7. Kernel derefs system call to
0x1 and executes shellcode
located there.

ROOT!

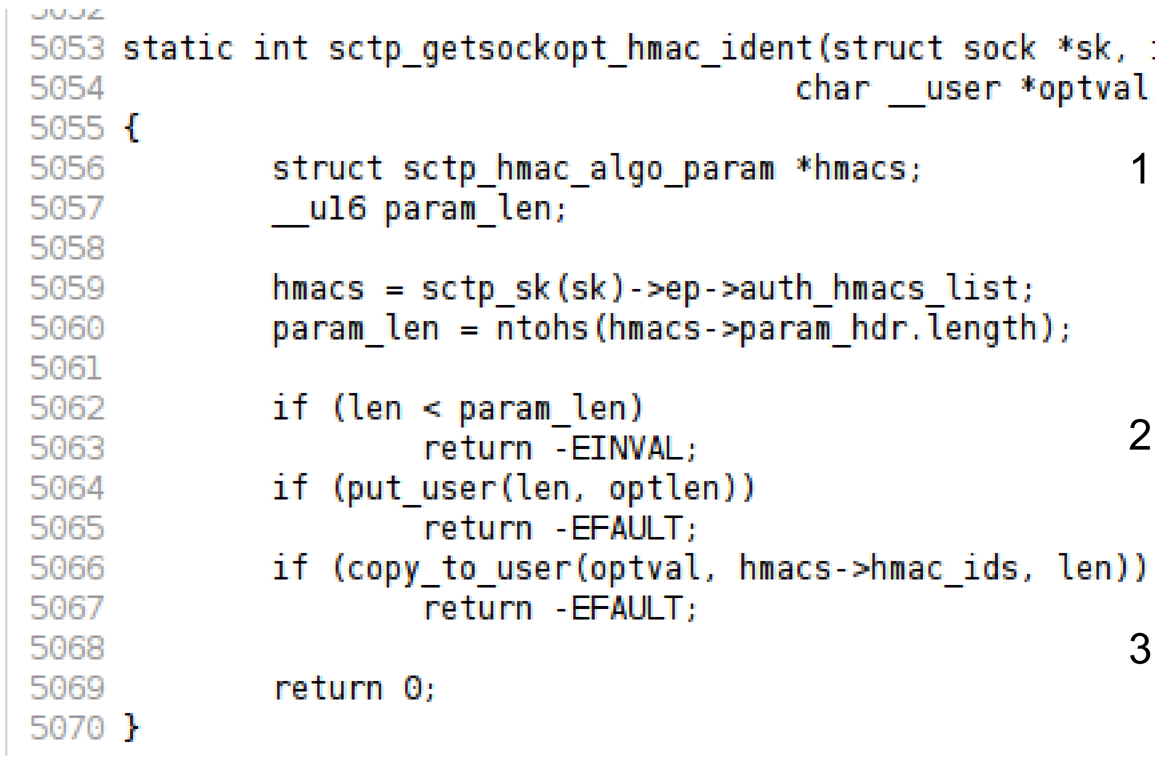
Protection Against User Derefs

- /proc/sys/vm/mmap_min_addr
  - Can't mmap to addrs < mmap_min_addr
  - Blocks NULL ptr derefs, but not all user derefs
  - Has been bypassed multiple times [17]
- UDEREF in grsecurity [18]
  - Segmentation on i386
  - Just released x64 support (spender/pipacs) [19]

Memory Disclosure

- Leak sensitive kernel memory to userspace
  - Commonly through unbounded copy_to_user()
- Pair up info leak w/setuid exploit
  - Leak task_struct/vma information for ASLR bypass
- Pair up info leak w/kernel exploit
  - Leak stack canary value from task_struct

Memory Disclosure Example



```
static int sctp_getsockopt_hmac_ident(struct sock *sk, int len,
                                    char __user *optval, int __user *optlen)
{
        struct sctp_hmac_algo_param *hmacs;
        __u16 param_len;

        hmacs = sctp_sk(sk)->ep->auth_hmacs_list;
        param_len = ntohs(hmacs->param_hdr.length);

        if (len < param_len)
                return -EINVAL;
        if (put_user(len, optlen))
                return -EFAULT;
        if (copy_to_user(optval, hmacs->hmac_ids, len))
                return -EFAULT;

        return 0;
}
```

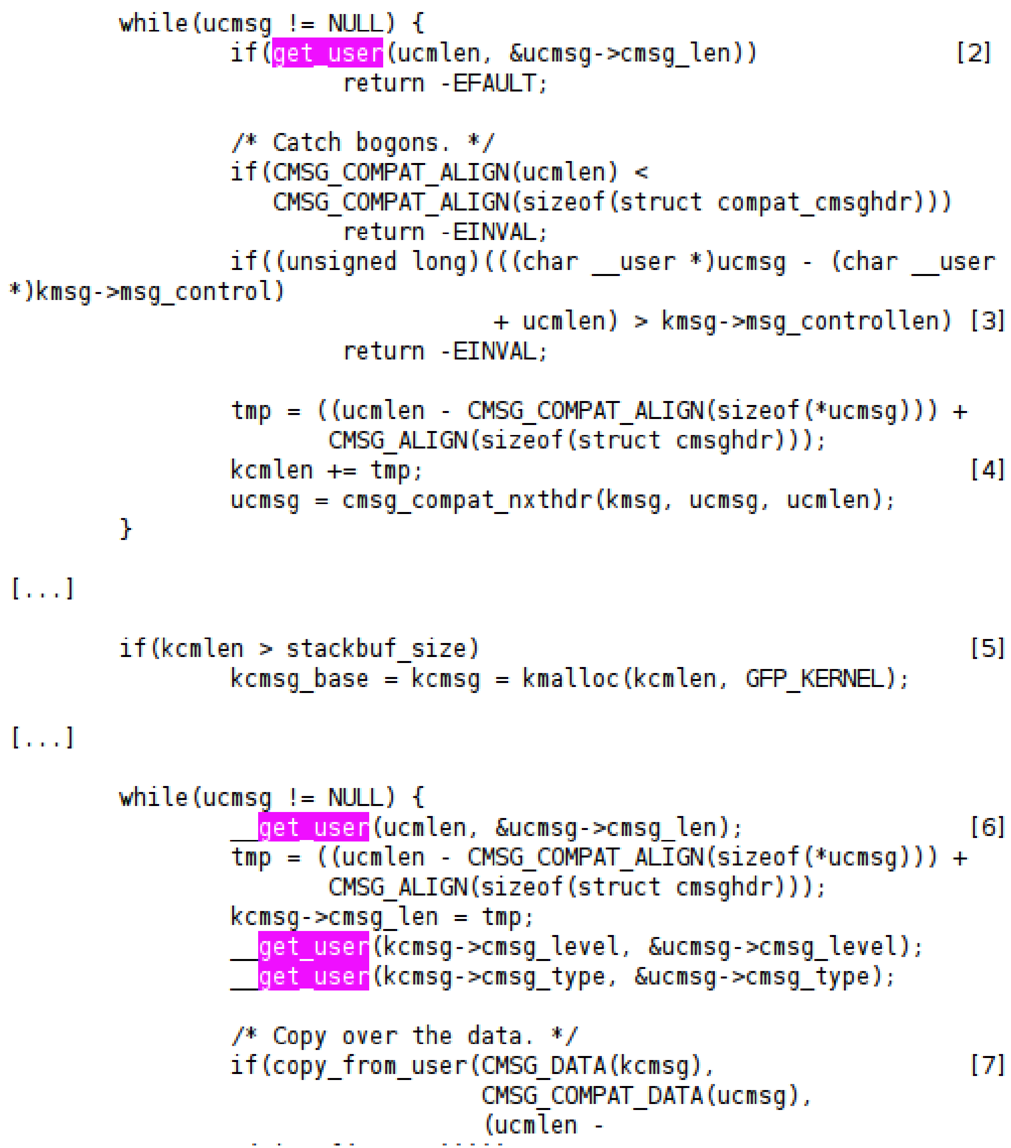
1. len is attacker controlled
2. no upper bounds check
3. unbounded copy_to_user

SCTP getsockopt() kernel memory disclosure [20]

Race Conditions

- Comes in many forms:
  - eg. ptrace_attach, user copies
- Hard on UP, easier on SMP
  - Force kernel to sleep/reschedule on UP
  - Great work from sgrakkyu and twiz [21]
- Ex: sendmsg() multiple user copy vuln

Race Condition Example



```
        while(ucmsg != NULL) {
                if(get_user(ucmlen, &ucmsg->cmsg_len))              [2]
                        return -EFAULT;

                /* Catch bogons. */
                if(CMSG_COMPAT_ALIGN(ucmlen) <
                   CMSG_COMPAT_ALIGN(sizeof(struct compat_cmsghdr)))
                        return -EINVAL;
                if((unsigned long)(((char __user *)ucmsg - (char __user
*)kmsg->msg_control)
                                   + ucmlen) > kmsg->msg_controllen) [3]
                        return -EINVAL;

                tmp = ((ucmlen - CMSG_COMPAT_ALIGN(sizeof(*ucmsg))) +
                       CMSG_ALIGN(sizeof(struct cmsghdr)));
                kcmlen += tmp;                                       [4]
                ucmsg = cmsg_compat_nxthdr(kmsg, ucmsg, ucmlen);
        }

[...]

        if(kcmlen > stackbuf_size)                                   [5]
                kcmsg_base = kcmsg = kmalloc(kcmlen, GFP_KERNEL);

[...]

        while(ucmsg != NULL) {
                __get_user(ucmlen, &ucmsg->cmsg_len);                [6]
                tmp = ((ucmlen - CMSG_COMPAT_ALIGN(sizeof(*ucmsg))) +
                       CMSG_ALIGN(sizeof(struct cmsghdr)));
                kcmsg->cmsg_len = tmp;
                __get_user(kcmsg->cmsg_level, &ucmsg->cmsg_level);
                __get_user(kcmsg->cmsg_type, &ucmsg->cmsg_type);

                /* Copy over the data. */
                if(copy_from_user(CMSG_DATA(kcmsg),                  [7]
                                  CMSG_COMPAT_DATA(ucmsg),
                                  (ucmlen -
```

[2] copy length value from userspace

[3] sanity check length value

RACE WINDOW!

[6] re-copy length value from userspace

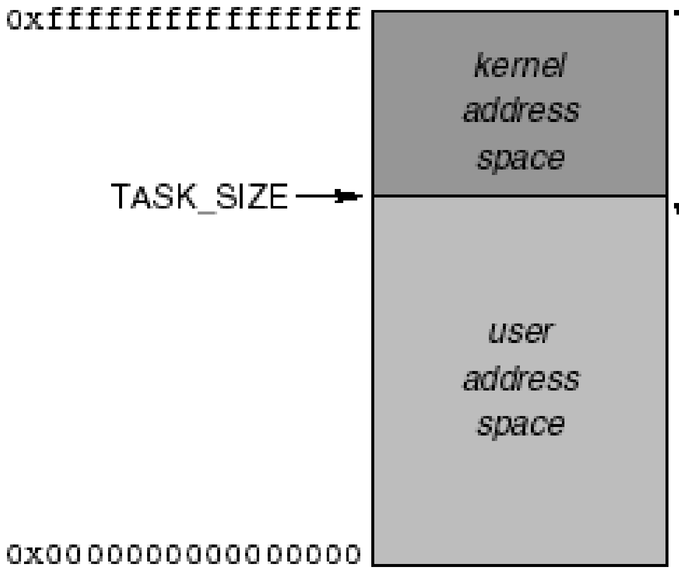
[7] perform copy_from_user with length value

MM Desynchronization

- Confusing the MM subsystem
  - Desynchronize book-keeping state of MM from what the actual state is in memory
- PS3 hack analogy [22]
  - George Hotz, OS ↔ hypervisor
  - Glitching the memory bus during memory mappings
- We can do the same for the kernel MM
  - But of course between userspace ↔ OS
  - Our “glitching” is just leveraging a book-keeping bug

MM Desynchronization

- Some classic exploits from isec.pl guys
  - mremap/unmap/do_brk [23,24,25]
- Hugely simplified do_brk
  - Expand heap via brk(2)
  - Lack of addr sanity checking
  - Expand past TASK_SIZE
  - mprotect() to alter MMU prot
  - Kernel confused..thinks it's part of heap, approves!
  - Kernel memory now writable by userspace



Agenda

- Why the Linux Kernel?
- A History of Vulns
- Vulnerability Classes
- **Wrap-up**

What You Can Do!

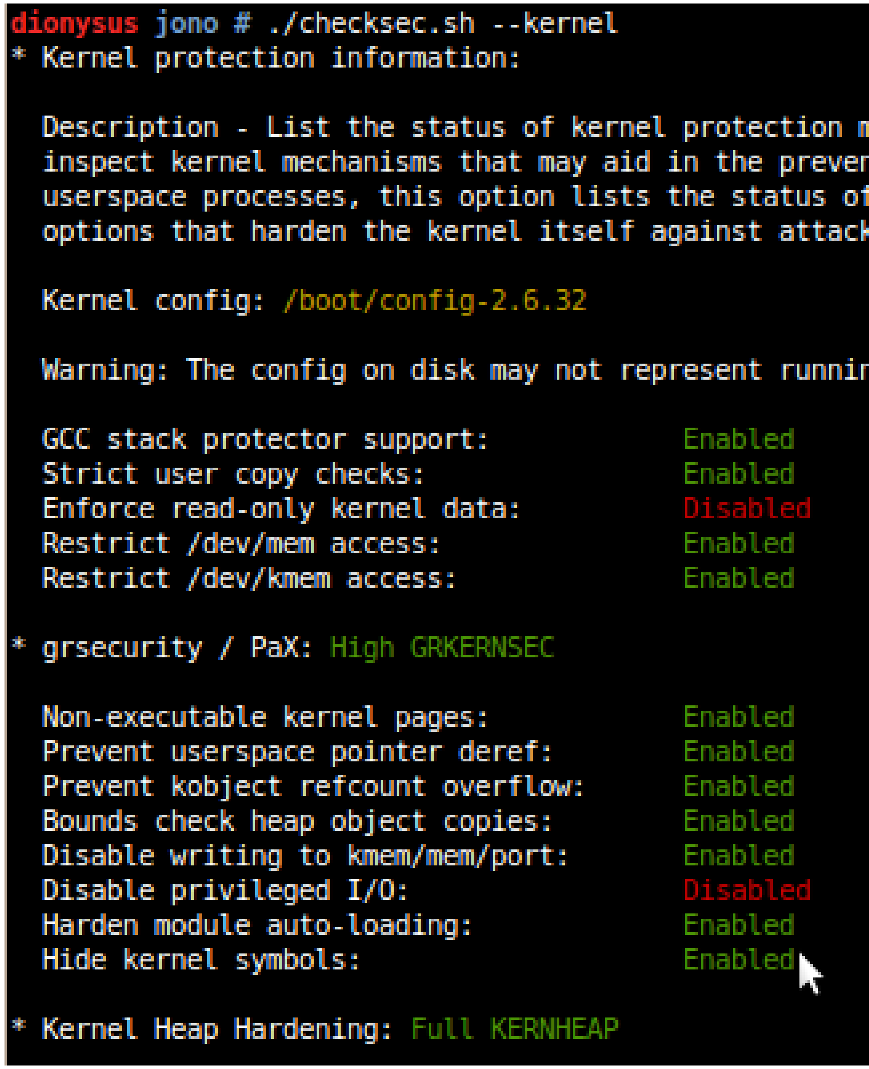
- Researchers:
  - Spend more time in kernel space!
  - There's much fun to be had!
- Administrators
  - Distros are conservative, poke them!
  - Lots of hardening you can do on your own
  - grsecurity / PaX / KERNHEAP patchsets [26,14]
  - Most importantly, support/sponsor these guys for their hard work

Auditing With checksec.sh

- checksec.sh
  - Tobias Klein [27]
  - ASLR, RELRO, NX, PIE, canaries, etc
- Now with kernel support!
  - Checks for a number of kernel hardening features

Patched version w/kernel support:

http://jon.oberheide.org/files/checksec.sh



```
dionysus jono # ./checksec.sh --kernel
* Kernel protection information:

  Description - List the status of kernel protection m
  inspect kernel mechanisms that may aid in the preven
  userspace processes, this option lists the status of
  options that harden the kernel itself against attack

  Kernel config: /boot/config-2.6.32

  Warning: The config on disk may not represent runnin

  GCC stack protector support:              Enabled
  Strict user copy checks:                  Enabled
  Enforce read-only kernel data:            Disabled
  Restrict /dev/mem access:                 Enabled
  Restrict /dev/kmem access:                Enabled

* grsecurity / PaX: High GRKERNSEC

  Non-executable kernel pages:              Enabled
  Prevent userspace pointer deref:          Enabled
  Prevent kobject refcount overflow:        Enabled
  Bounds check heap object copies:          Enabled
  Disable writing to kmem/mem/port:         Enabled
  Disable privileged I/O:                   Disabled
  Harden module auto-loading:               Enabled
  Hide kernel symbols:                      Enabled

* Kernel Heap Hardening: Full KERNHEAP
```

Take Away

- Message is ***not***: “Don't use Linux, it's insecure, lolz!”
- Security is not measured in absolutes
  - Risk management → uncertainty management

“There are known knowns. There are things we know that we know. There are known unknowns. That is to say there are things that we now know we don't know. But there are also unknown unknowns. There are things we do not know we don't know.”

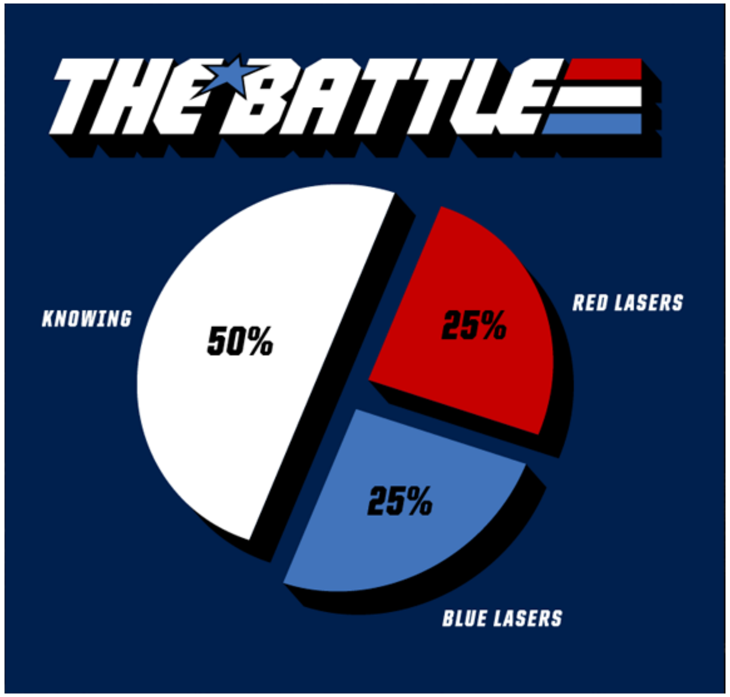
-- Donald Rumsfeld

Or, more concisely:

“Now you know, and knowing is half the battle!” -- GI JOE

Thank you

QUESTIONS?



Jon Oberheide / @jonoberheide / jono@sciosecurity.com

References

[1] http://zenthought.org/content/project/flashrec
[2] http://www.grsecurity.net/~spender/wunderbar_emporium.tgz
[3] http://c-skills.blogspot.com/2009/04/udev-trickery-cve-2009-1185-and-cve.html
[4] http://jon.oberheide.org/files/cve-2009-1185.c
[5] http://sgrakkyu.antifork.org/sctp_houdini.c
[6] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-6332
[7] http://www.springerlink.com/content/m2783777l47t3836/
[8] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1146
[9] http://lwn.net/Articles/342182/
[10] http://www.grsecurity.net/~spender/fortify_source_30_percent_coverage_sucks.patch
[11] http://gcc.gnu.org/onlinedocs/gcc/Object-Size-Checking.html
[12] http://sgrakkyu.antifork.org/tiocl_houdini.c
[13] http://www.phrack.org/issues.html?issue=66&id=15
[14] http://www.subreption.com/kernheap/
[15] http://www.grsecurity.net/~spender/enlightenment.tgz
[16] http://marc.info/?l=openbsd-cvs&m=125676466108709&w=2
[17] http://blog.cr0.org/2009/06/bypassing-linux-null-pointer.html
[18] http://www.grsecurity.net/~spender/uderef.txt
[19] http://www.grsecurity.net/pipermail/grsecurity/2010-April/001024.html
[20] http://jon.oberheide.org/files/cve-2008-4113.c
[21] http://www.phrack.org/issues.html?issue=64&id=6
[22] http://rdist.root.org/2010/01/27/how-the-ps3-hypervisor-was-hacked/
[23] http://isec.pl/vulnerabilities/isec-0012-do_brk.txt
[24] http://isec.pl/vulnerabilities/isec-0013-mremap.txt
[25] http://isec.pl/vulnerabilities/isec-0014-mremap-unmap.txt
[26] http://grsecurity.net/features.php
[27] http://www.trapkit.de/tools/checksec.html