Dirty CMOS Tricks

Or, My Rootkit Is Better Than Your Rootkit

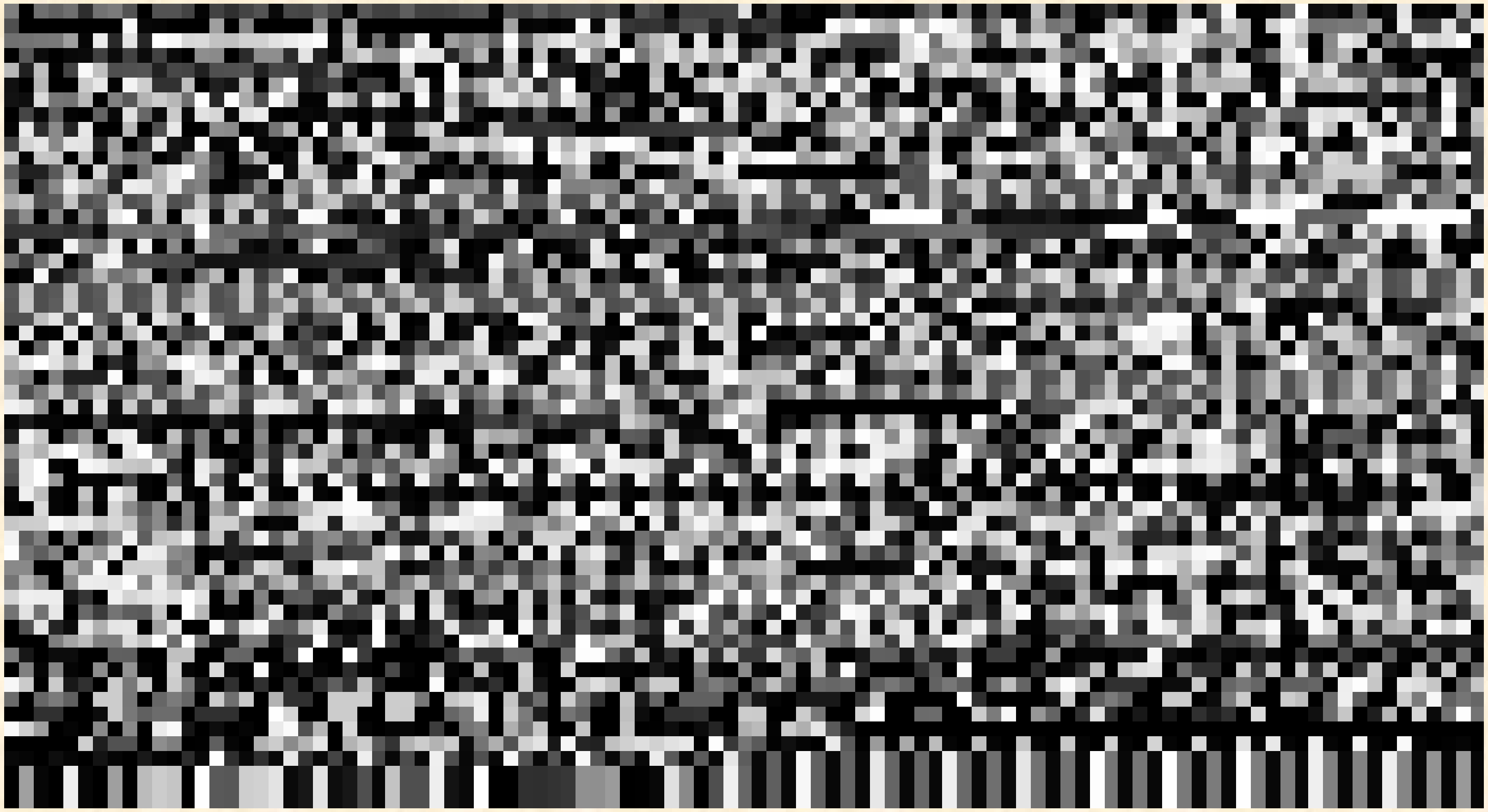
Me

- Network Security Consultant
- Doxpara Research

PCs

- IBM PC introduced in September 1981
- 16-bit I/O port address space
- 16KB BIOS ROM (date: 4/24/1981)
- Monolithic BIOS

© 1981, IBM



No, I Meant The Real PCs

- IBM PC/XT introduced June, 1984
- Now with fixed disks
- Modular BIOS
- And an I/O mapped RTC: the Motorola MC146818

CMOS RAM

- RTC contained 64 or 128 bytes of battery-backed storage
- Contains equipment info used by POST and DOS

Magic Numbers

```
;----------------------------------------
;       CMOS EQUATES FOR THIS SYSTEM    ;
;-----------------------------------------------------------------------------

CMOS_PORT       EQU    070H          ; I/O ADDRESS OF CMOS ADDRESS PORT

CMOS_DATA       EQU    071H          ; I/O ADDRESS OF CMOS DATA PORT
```

Fast Forward

- Clones (Compaq and friends)
- PC/AT and the 286’s 24 bit protected mode
- “Let’s wire the keyboard controller to the reset pin”
- IBM’s PS/2 line and MCA
- Netware
- DOS
- Lotus 1-2-3, Autocad, Wordperfect, etc

Keep Going

- 386 and 32-bit protected mode
- Windows
- OS/2
- Windows NT
- Superscalar architectures
- Microsoft’s licensing practices

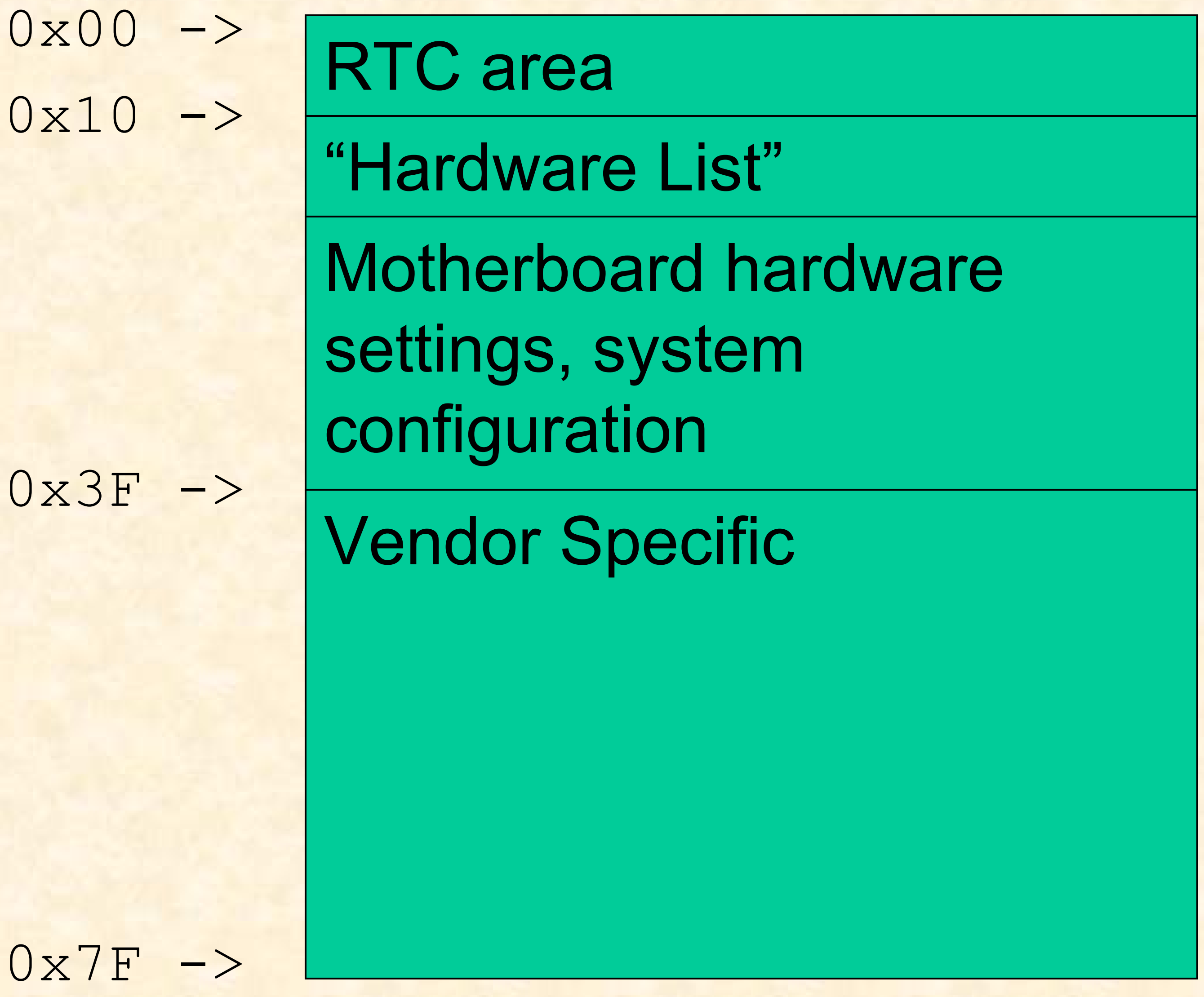
Now

- Lots of Win9x, Win2k
- Opteron / Athlon 64 soon
- They all have a CMOS data area of some kind

What’s in the CMOS?

- Region 0x10-0x3F almost universally similar (but not PS/2s of course)
- Checksum for first 64 bytes
- Checksum introduced for second 64 bytes once 128 byte RTCs became available

CMOS Map


0x00 ->
0x10 ->
0x3F ->
0x7F ->
RTC area
“Hardware List”
Motherboard hardware settings, system configuration
Vendor Specific

Talking to the CMOS

- Win9x: peek and poke all you want (DEBUG.COM)
- Win2k: need ring 0 driver
- Linux: iopl()
- FreeBSD: open /dev/io
- OpenBSD: i386_iopl()

- “**WARNING** You can really hose your machine if you enable user-level I/O and write to hardware ports without care.” – OpenBSD i386_iopl(3) page

Finding the BIOS

- FreeBSD: bios_sigsearch()
- Everyone else: scan 0xc0000 through 0xffff0

Reverse Engineering (1)

- `/dev/mem`
- `\Device\PhysicalMemory`
- `MOV`
- `MOVQ` (MMX)
- `MOVDQ` (SSE)

Reverse Engineering (2)

- 0xAA55
- "¬U"?
- Not U either
- 0xFFFF0 (FFFF:000F) (1048560 dec)

Defending (1)

- BIOS backups
- Dual-BIOS motherboards
- Runtime BIOS and CMOS IDS checksums
- DENY PHYSICAL ACCESS
- Create and test BIOS reload procedures when you do server reload-from-backup testing

Defending (2)

- Consider complexity of BIOS reload during hardware vendor evaluation
- Jumpers…
- Emergency reload floppies taped to the servers w/ secured backups and tested regularly or at least during post-install

Defending (3)

- Deny I/O port access from userland
- Require driver signatures in NT
- Event IDs 577 + 578 = bad, remote logging = good
- BSD's securelevel
- strip -N iopl vmlinux FIXME replace with ret

Dirty Tricks

- Set the boot password (ransom it!)
- Clear the boot password on kiosks
- Set a random boot password (deadly to laptops)

Really Dirty Tricks

- Overclocking is fun
- Change the boot logo
- Change the big boot logo too

Really Really Dirty Tricks

- Reset the RTC to the date of the OS image, THEN install your rootkit (change it back after)
- Boot off of alternative media to escape OS limitations if you can’t get kernel access

You Are A Bad Bad Person

- Boot off of the network!
- PXE is your friend
- Boot ALL the systems off the network
- Knoppix, LTSP, NFS root HOWTO
- Demoscene programs

Am I Fired Yet?

Thanks

- Ralf Brown’s Interrupt List
- Mastodonic’s IBM Vintage Personal Computers Page
- IBM
- ImageMagick for convert
- Dan Kaminsky for hardware

And Now For Something Completely Different…