VENOM

CVE-2015-3456

Jason Geffner

Principal Security Researcher

VULNERABILITY NAMING

A ROSE BY ANY OTHER NAME…

PURPOSE OF NAMING

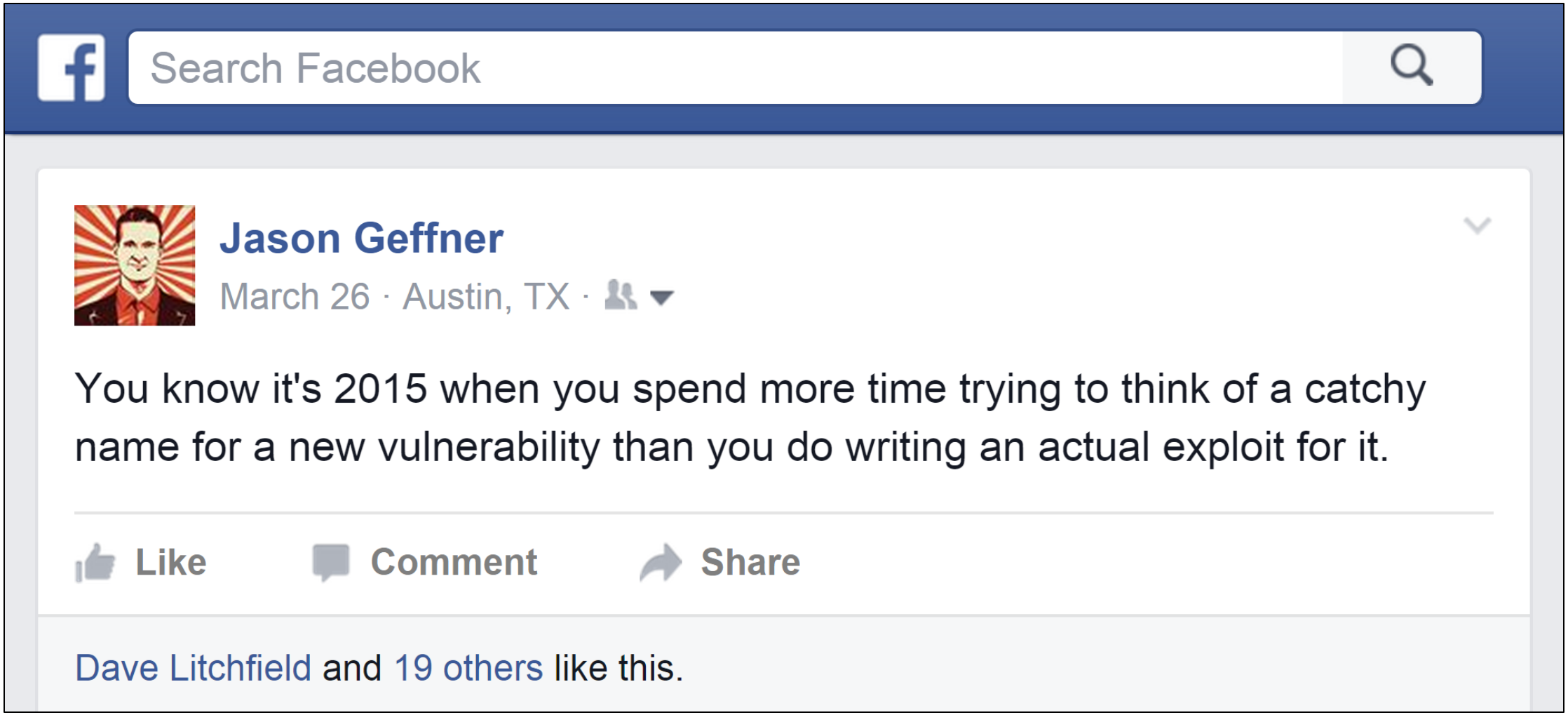
http://oceanservice.noaa.gov/facts/storm-names.html

NAMED VULNERABILITIES

CVE Number	Name
CVE-2011-3389	BEAST
CVE-2013-3587	BREACH
CVE-2014-0160	Heartbleed
CVE-2014-1568	BERserk
CVE-2014-3566	POODLE
CVE-2014-4114	Sandworm

CVE Number	Name
CVE-2014-6271	Shellshock
CVE-2014-6321	WinShock
CVE-2015-0204	FREAK
CVE-2015-0235	GHOST
CVE-2015-1130	Rootpipe
CVE-2015-3456	VENOM

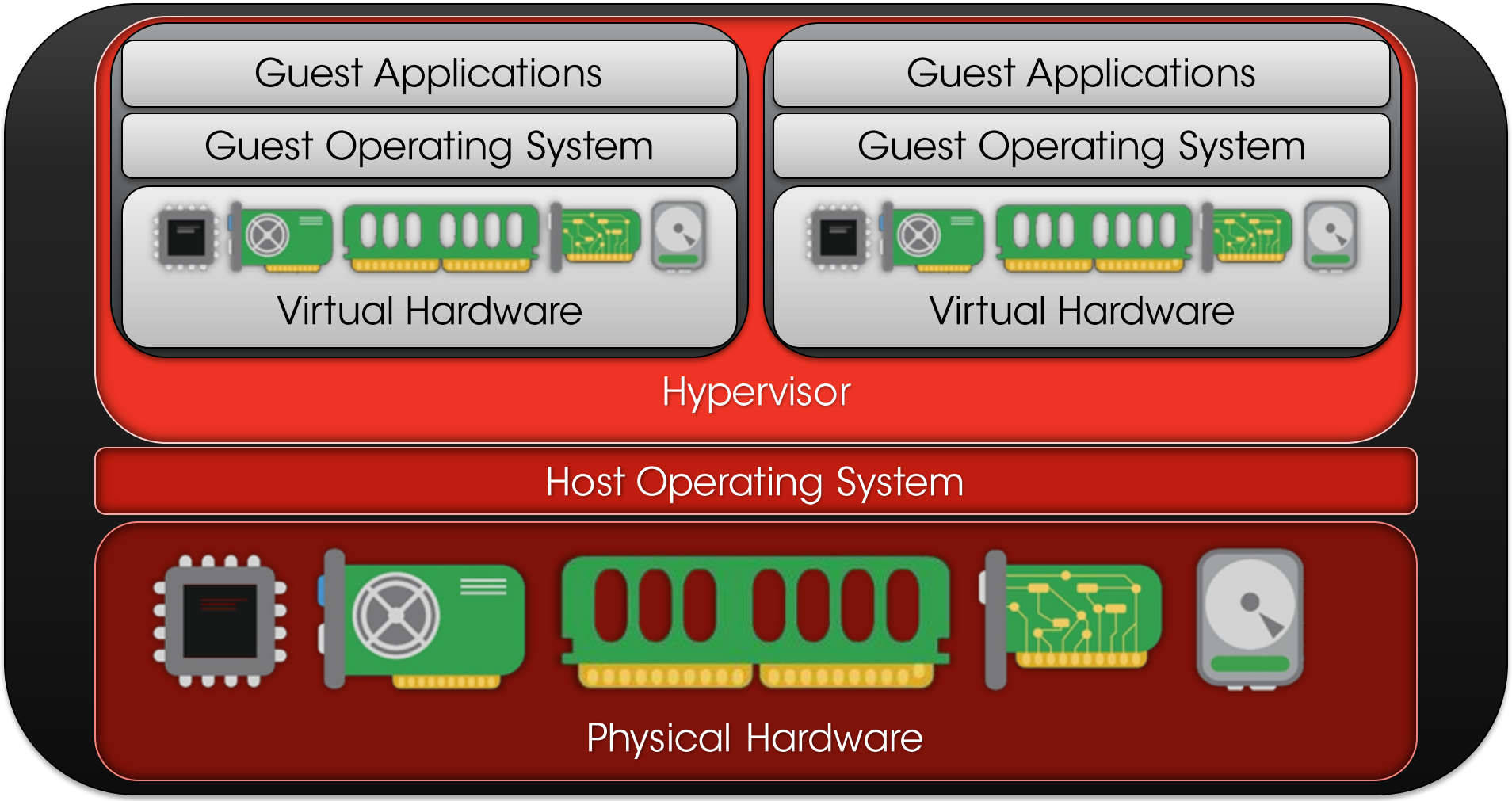
Virtualized Environment Neglected Operations Manipulation



VIRTUALIZATION

HOW DOES A VM WORK?

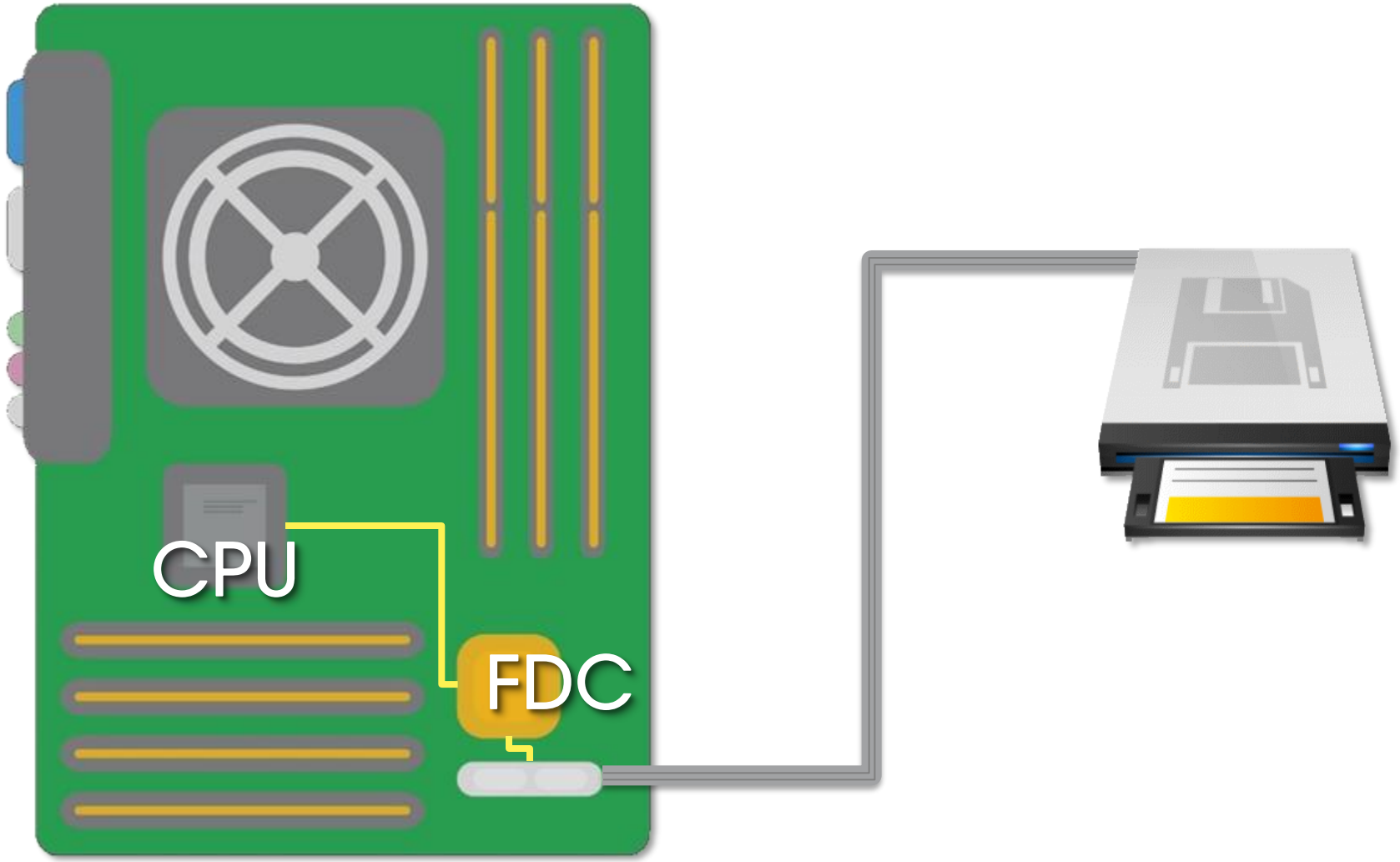
WHAT IS VIRTUALIZATION?



FLOPPY DISK CONTROLLER

YOU DOWN WITH FDC? YEAH YOU KNOW ME!

FLOPPY DISK CONTROLLER



1.0 INTRODUCTION

The 82078 (44 pin) enhanced floppy disk controller incorporates several new features allowing for easy implementation in both the portable and desktop markets. It provides a low cost, small form factor solution targeted for 5.0V and 3.3V platforms that do not require more than two drive support.

The 82078 (44 pin) implements these new features while remaining functionally compatible with 82077SL/82077AA/8272A floppy disk controllers.

Together with a 24 MHz crystal, a resistor package and a device chip select, these devices allow for the most integrated solution available. The integrated analog PLL data separator has better performance than most board level discrete PLL implementations and can be operated at 1 Mbps/500 Kbps/300 Kbps/250 Kbps. A 16-byte FIFO substantially improves system performance especially in multi-master systems (e.g. Microchannel, EISA).

Figure 1-1 is a block diagram of the 82078.

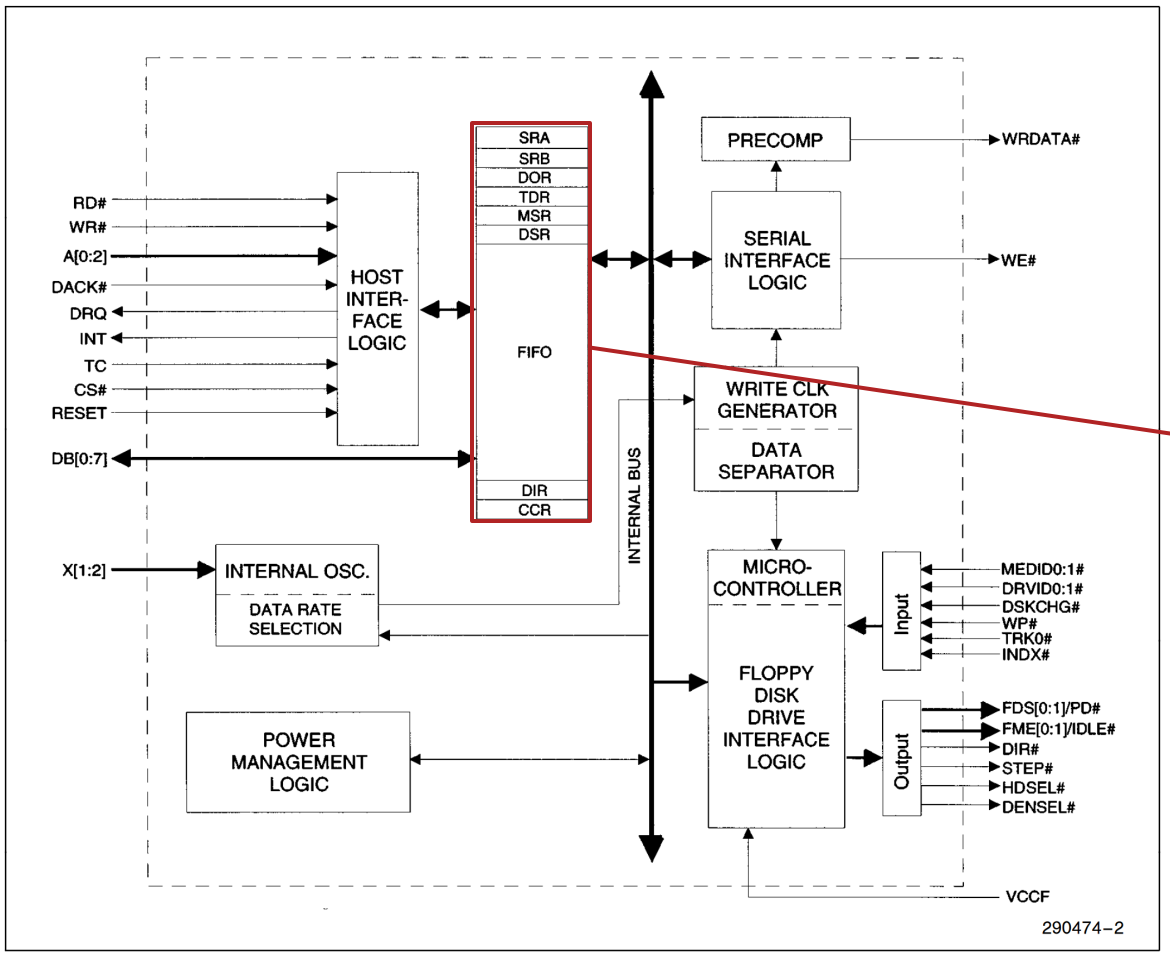
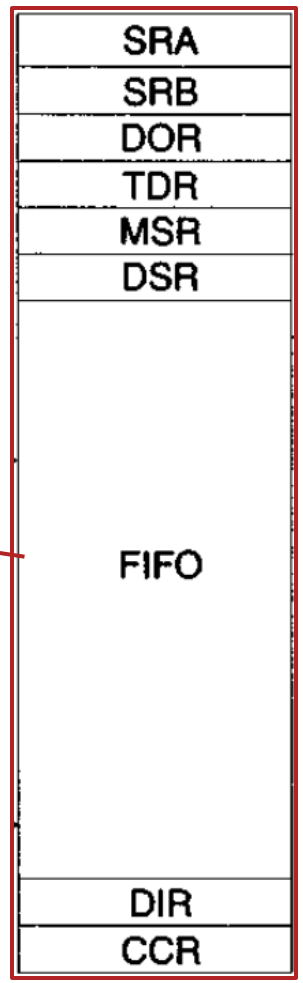
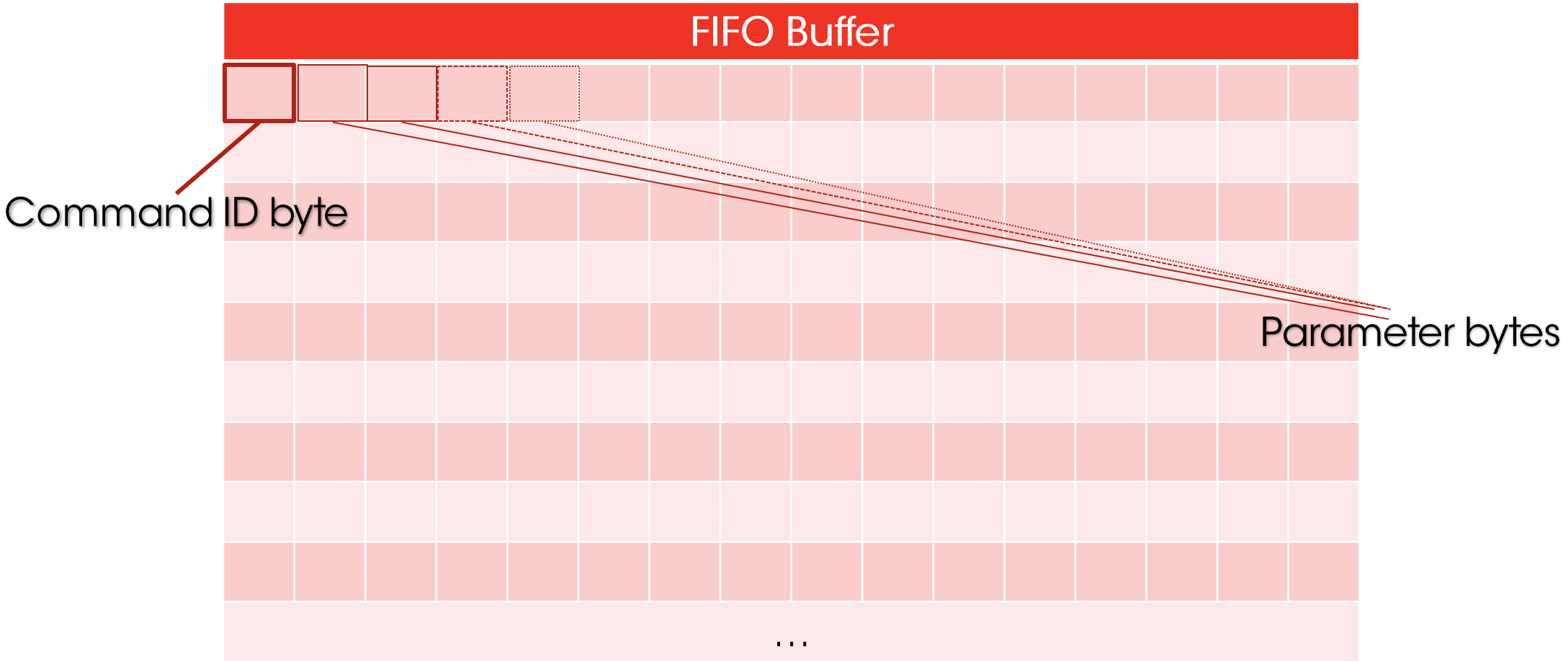


Figure 1-1. 82078 Block Diagram

300 Kbps/250 Kbps. A 16-byte FIFO substantially improves system performance especially in multi-master systems (e.g. Microchannel, EISA).



SENDING COMMANDS TO FDC



THE VULNERABILITY

CVE-2015-3456

QEMU'S FDC COMMAND HANDLERS

```
static const struct {
    uint8_t value;
    uint8_t mask;
    const char* name;
    int parameters;
    void (*handler)(FDCtrl *fdctrl, int direction);
    int direction;
} handlers[] = {
    { FD_CMD_READ,  0x1f, "READ",  8, fdctrl_start_transfer, FD_DIR_READ },
    { FD_CMD_WRITE, 0x3f, "WRITE", 8, fdctrl_start_transfer, FD_DIR_WRITE },
    { FD_CMD_SEEK,  0xff, "SEEK",  2, fdctrl_handle_seek },
…
};
```

QEMU'S FDC COMMAND LOOKUP

```
if (fdctrl->data_pos == 0) {
    /* Command */
    pos = command_to_handler[value & 0xff];
    FLOPPY_DPRINTF("%s command\n", handlers[pos].name);
    fdctrl->data_len = handlers[pos].parameters + 1;
    fdctrl->msr |= FD_MSR_CMDBUSY;
}
```

THE VULNERABILITY

```
fdctrl->fifo[fdctrl->data_pos++] = value;

if (fdctrl->data_pos == fdctrl->data_len) {
    /* We now have all parameters
     * and will be able to treat the command
     */
…
    pos = command_to_handler[fdctrl->fifo[0] & 0xff];
    FLOPPY_DPRINTF("treat %s command\n", handlers[pos].name);
    (*handlers[pos].handler)(fdctrl, handlers[pos].direction);
}
```

“READ ID” ATTACK VECTOR

```
static void fdctrl_handle_readid(FDCtrl *fdctrl, int direction)
{
    FDrive *cur_drv = get_cur_drv(fdctrl);

    cur_drv->head = (fdctrl->fifo[1] >> 2) & 1;
    timer_mod(fdctrl->result_timer,
              qemu_clock_get_ns(QEMU_CLOCK_VIRTUAL) +
              (get_ticks_per_sec() / 50));
}
```

“DRIVE SPECIFICATION COMMAND” ATTACK VECTOR

```
static void fdctrl_handle_drive_specification_command(FDCtrl *fdctrl, int direction)
{
    FDrive *cur_drv = get_cur_drv(fdctrl);
    if (fdctrl->fifo[fdctrl->data_pos - 1] & 0x80) {
        if (fdctrl->fifo[fdctrl->data_pos - 1] & 0x40) {
            fdctrl->fifo[0] = fdctrl->fifo[1];
            fdctrl->fifo[2] = 0;
            fdctrl->fifo[3] = 0;
            fdctrl_set_fifo(fdctrl, 4);
        } else {
            fdctrl_reset_fifo(fdctrl);
        }
    } else if (fdctrl->data_len > 7) {
        fdctrl->fifo[0] = 0x80 |
            (cur_drv->head << 2) | GET_CUR_DRV(fdctrl);
        fdctrl_set_fifo(fdctrl, 1);
    }
}
```

PROBING

KNOCK-KNOCK…

PROBING METHOD

After each step below, we read FDC registers and drain FIFO buffer

1. Boot system
2. Send software reset to FDC if FIFO not ready for data transfer
3. Write `FD_CMD_READ_ID` to FIFO
4. Write parameter byte to FIFO
5. Sleep for one second

Timing the `result_timer`:

1. Write `FD_CMD_READ_ID` and parameter byte to FIFO
2. Wait until data is available to read from the FIFO

PROBING RESULTS

Platform	Initial Status						After `FD_CMD_READ_ID`						After one second													Timer (ms)
	SRA	SRB	DOR	TDR	MSR	DIR	SRA	SRB	DOR	TDR	MSR	DIR	SRA	SRB	DOR	TDR	MSR	DIR	FIFO							
QEMU	FF	C0	0C	00	80	80	FF	C0	0C	00	90	80	FF	C0	0C	00	D0	80	00	00	00	00	00	02	02	20.81
VirtualBox	FF	C0	0C	00	80	00	FF	C0	0C	00	90	00	FF	C0	0C	00	D0	00	40	05	01	00	00	01	02	20.32
Amazon	FF	C0	0C	00	80	00	FF	C0	0C	00	80	00	FF	C0	0C	00	D0	00	00	00	00	00	00	01	02	20.28
Rackspace	FF	C0	0C	00	80	80	FF	C0	0C	00	80	80	FF	C0	0C	00	D0	80	00	00	00	00	00	01	02	20.29
Bochs	FF	FF	0C	20	80	7F	FF	FF	0C	20	D0	7F														
VMware	FF	FF	0C	FF	80	00	FF	FF	0C	FF	D0	00														
Azure	FF	FC	1C	FF	80	FF	FF	FC	1C	FF	D0	FF														
Google	FF	FF	FF	FF	FF	FF																				

No FDC

`FD_MSR_DIO` bit set:
FDC wrote output data to FIFO
before receiving parameter byte

OPTIONS

BLACK HAT OR WHITE HAT

WHAT CAN WE DO WITH THIS VULNERABILITY?

COORDINATED DISCLOSURE

EVERYONE WANTS TO BE THE FIRST TO KNOW

How do I do coordinated disclosure?

Web Maps Shopping Images News More Search tools

Your search - **How do I do coordinated disclosure?** - did not match any documents.

Suggestions:

- Make sure all words are spelled correctly.
- Try different keywords.
- Try more general keywords.

DAN KAMINSKY

- One of the few people who've had to deal with a large-scale coordinated disclosure

- Recommended starting by going to the biggest players

PRE-DISCLOSURE

- Spoke with security teams at major companies affected by VENOM
- Was put in touch with QEMU's Project Leader

- Pre-disclosed vulnerability details to software vendors:

Operating System Distribution Security Mailing List

OPERATING SYSTEM DISTRIBUTION SECURITY MAILING LIST

- ALT Linux
- Amazon Linux AMI
- Arch Linux
- Chrome OS
- Debian
- FreeBSD
- Gentoo
- MontaVista Software
- NetBSD
- Openwall
- Oracle
- Red Hat
- Slackware
- SUSE
- Ubuntu
- Wind River

SAD FACTS

- Fact #1

  Everyone wants to get the pre-disclosure details before it's disclosed publicly.

- Fact #2

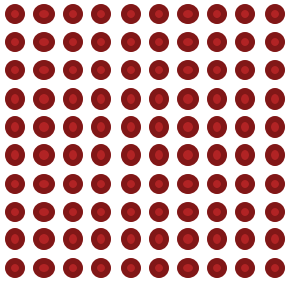
  You can't include everyone on the pre-disclosure list, so you're going to have people angry with you no matter what.


linode
Joyent
ubuntu
DigitalOcean
f5
FORTINET
rackspace
amazon
web services™
ORACLE
SUSE
IBM
FireEye
hp
debian
CentOS
QEMU
CISCO
redhat
CITRIX
Xen
Project
UpCloud
liquidweb
Barracuda

MEDIA RESPONSE

THEY SEE ME ROLLIN…

PRESS ACCURACY



100+ Number of news articles published on VENOM

6 Number of reporters with whom I spoke

1 Number of reporters who asked me to review accuracy of their articles

VENOM - A New Computer Virus Poisoning The Virtual World

May 23, 2015

Venom, bigger than Heartbleed, affects all datacenters as a zero day bug

05.14.15

"VENOM" FUD Attack — Like "Heartbleed" FUD Attack — Linked to Microsoft

VULNERABILITY RESEARCH ADVICE

FINDING THE NEXT VENOM

FALLACY OF LINUS'S LAW

"Given enough eyeballs, all bugs are shallow"

- The **quality** of the eyeballs is much more important than the **quantity** of the eyeballs

- Until **your** eyeballs have reviewed something, don't assume that all bugs have been found

FINDING A NOTEWORTHY TARGET

- 8,000 new CVEs last year = 22 new CVEs per day

- Differentiate by focusing on major trends
  - Mobile devices
  - Internet of Things
  - Cloud
  - Security products and technologies
  - Big data
  - Automotive technologies
  - Healthcare technologies
  - Entertainment technologies

- Look at well-known products and technologies used by most consumers or by big name enterprises
- Remotely exploitable vulnerabilities are much more interesting than local

DISCOVERING THE NEXT BIG VULNERABILITY

- Fuzzing
  - Everything has already been fuzzed
  - Unless you have a new (and genius) approach to fuzzing, don't bother fuzzing old technologies
- Automated static program analysis gets better every year, but…
  - Is far from perfect
  - Is too often relied upon instead of manual code review
- Manual code review
  - Tedious ⇒ rarely done comprehensively ⇒ how the juicy bugs can be found

Q & A

CROWDSTRIKE