Failure is not an option*

A journey through software bugs

Philippe Biondi

Nov 20th 2015 / GreHack

AIRBUS GROUP

Outline

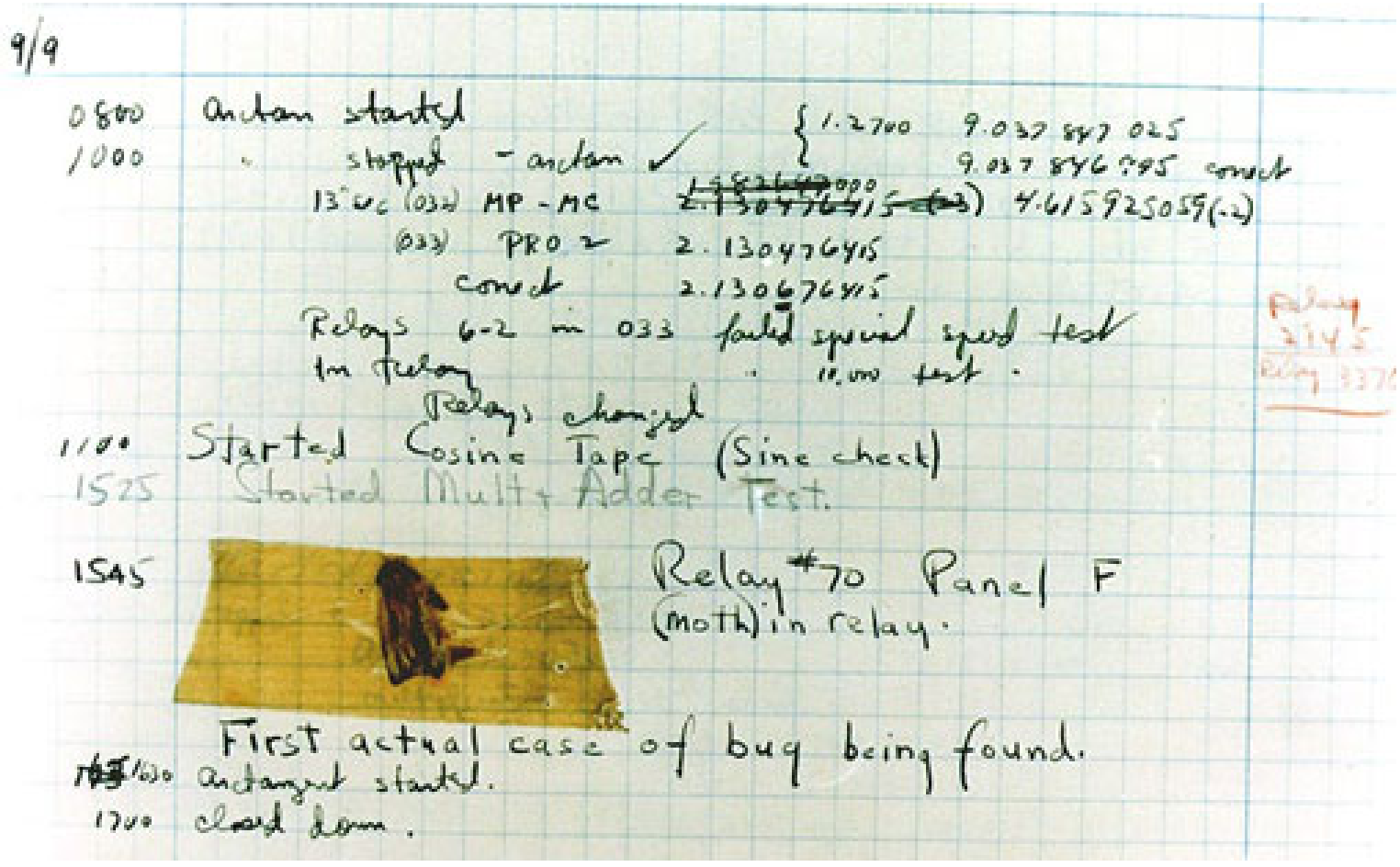
1. Bugs!
2. Avoiding and Finding bugs
3. Bugs still happen
4. Why do bugs still happen ?!
5. Living with bugs

Outline

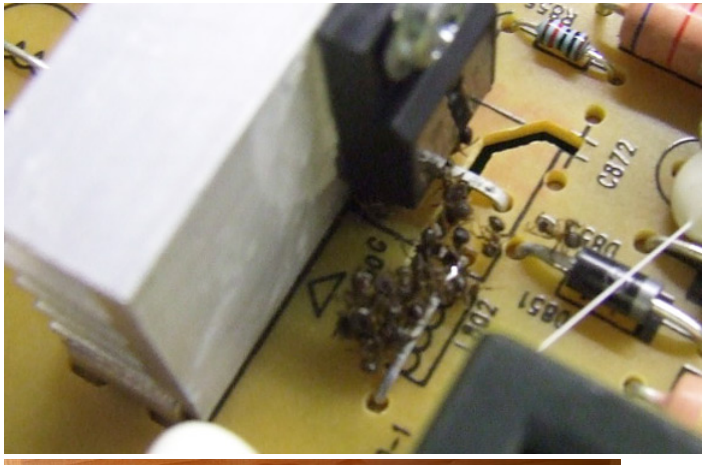
1. Bugs!
2. Avoiding and Finding bugs
3. Bugs still happen
4. Why do bugs still happen ?!
5. Living with bugs

The ancestor of all bugs

Moth in relay



Still nowadays[1]



[1] http://www.theregister.co.uk/2010/11/26/ventblockers_2/

Valve's Steam on Linux[2]

Steam can clean your home and more

```
STEAMROOT="$(cd "${0%/*}" && echo $PWD)"

# Scary!
rm -rf "$STEAMROOT/"*
```

[2] https://github.com/valvesoftware/steam-for-linux/issues/3671

Haunted doors[3]

- Office doors are keycard-protected
- Doors were slow to open : 5 to 30s, sometimes more
- Everyone had his ninja techniques that seemed to open them faster :
  - swipe card slowly
  - swipe card quickly
  - swipe once and wait
  - swipe furiously over and over until door unlocks
  - stand on one foot
  - etc.

[3] http://thedailywtf.com/articles/The-Haunted-Door

Haunted doors

- One day, an employee stayed late and alone in the office
- He heard clicks from doors being unlocked
- Eventually found the authentication server
- It turns out that:
  - log file was very big
  - it took a long time to open it and append a new line
  - all the card swipes were correctly queued
  - the software was still working on card swipes from the day before
  - problem was made even worse by people swiping multiple times

$\Longrightarrow$ door unlockings were not 30s long but $\approx$ **30h** long

$\Longrightarrow$ 30s was the time you had to wait for any door to open ; no need to swipe any card

Bad guys have bugs too

- Linux.Encoder.1 ransomware design flaw[4]
  - derives AES key and IV from libc `rand()`
  - seeded with current system timestamp

  $\Longrightarrow$ recover key from file's creation time

  $\Longrightarrow$ no need to pay the ransom!
- Power Worm ransomware variant[5]
  - author wanted to simplify his task: same AES key for all victims
  - ransomware encrypted files and did not store the key
  - programming error made the key actually random

  $\Longrightarrow$ no way to recover the files

---

[4] `http://labs.bitdefender.com/2015/11/linux-ransomware-debut-fails-on-predictable-encryption-key/`

[5] `http://news.softpedia.com/news/epic-fail-power-worm-ransomware-accidentally-destroys-victim-s-data-during-encryption-495.shtml`

RC4 implementation error

A bad implementation

```
int main(int argc, char *argv[]) {
        unsigned char S[256], c;
        unsigned char key[] = KEY;
        int klen = strlen(key);
        int i,j,k;
        /* Init S[] */
        for(i=0; i<256; i++)
                S[i] = i;
        /* Scramble S[] with the key */
        j = 0;
        for(i=0; i<256; i++) {
                j = (j+S[i]+key[i%klen]) % 256;
                S[i] ^= S[j];
                S[j] ^= S[i];
                S[i] ^= S[j];
        }
        /* Generate the keystream and cipher the input stream */
        i = j = 0;
        while (read(0, &c, 1) > 0) {
                i = (i+1) % 256;
                j = (j+S[i]) % 256;
                S[i] ^= S[j];
                S[j] ^= S[i];
                S[i] ^= S[j];
                c ^= S[(S[i]+S[j]) % 256];
                write(1, &c, 1);
        }
}
```

RC4 implementation error

A good implementation

```
int main(int argc, char *argv[]) {
        unsigned char S[256], c;
        unsigned char key[] = KEY;
        int klen = strlen(key);
        int i,j,k;
        /* Init S[] */
        for(i=0; i<256; i++)
                S[i] = i;
        /* Scramble S[] with the key */
        j = 0;
        for(i=0; i<256; i++) {
                j = (j+S[i]+key[i%klen]) % 256;
                k = S[i];
                S[i] = S[j];
                S[j] = k;
        }
        /* Generate the keystream and cipher the input stream */
        i = j = 0;
        while (read(0, &c, 1) > 0) {
                i = (i+1) % 256;
                j = (j+S[i]) % 256;
                k = S[i];
                S[i] = S[j];
                S[j] = k;
                c ^= S[(S[i]+S[j]) % 256];
                write(1, &c, 1);
        }
}
```

RC4 implementation error

Exchanging values

Classical way (using temporary variable)

```
tmp = a
a = b
b = tmp
```

To show-off

```
a = a+b
b = a-b
a = a-b
```

```
a += b
b = a-b
a -= b
```

```
a = a^b
b = a^b
a = a^b
```

```
a ^= b
b ^= a
a ^= b
```

RC4 implementation error

The bug

The working idiom

```
a = a^b
b = a^b
a = a^b
```

The buggy adaptation

```
S[i] = S[i]^S[j]
S[j] = S[i]^S[j]
S[i] = S[i]^S[j]
```

RC4 implementation error

The bug

- When i=j, we have

```
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
```

- i.e. actually

```
a = a^a
a = a^a
a = a^a
```

- $\Longrightarrow$ instead of exchanging a value with itself, we set it to 0
- $\Longrightarrow$ the RC4 state fills up with 0
- $\Longrightarrow$ the bitstream quickly degrades to a sequence of 0
- $\Longrightarrow$ encryption does not happen anymore

Beyond the code

Double-checked locking pattern does not work[6]

Single threaded version of a singleton instantiation

```
1  class Foo {
2    private Helper helper = null;
3    public Helper getHelper() {
4      if (helper == null)
5          helper = new Helper();
6      return helper;
7      }
8    // other functions and members...
9    }
```

[6] http://www.cs.umd.edu/~pugh/java/memoryModel/DoubleCheckedLocking.html

Beyond the code

Double-checked locking pattern does not work

Multithreaded version of a singleton instantiation

```
1  class Foo {
2    private Helper helper = null;
3    public synchronized Helper getHelper() {
4      if (helper == null)
5          helper = new Helper();
6      return helper;
7      }
8    // other functions and members...
9    }
```

Beyond the code

Double-checked locking pattern does not work

Multithreaded version of a singleton instantiation using the double-checked locking pattern.
Most calls to `getHelper()` will not be synchronized (better performance).

```
1   class Foo {
2   private Helper helper = null;
3   public Helper getHelper() {
4     if (helper == null)
5       synchronized(this) {
6         if (helper == null)
7           helper = new Helper();
8       }
9     return helper;
10    }
11  // other functions and members...
12  }
```

Beyond the code

Double-checked locking pattern does not work

Actual code that can be executed (after JIT)

```
1   call   01F6B210                    ; allocate space for Helper,
2                                      ; return result in eax
3   mov    dword ptr [ebp],eax         ; EBP is "helper" field. Store
4                                      ; the unconstructed object here.
5   mov    ecx,dword ptr [eax]         ; dereference the handle to
6                                      ; get the raw pointer
7   mov    dword ptr [ecx],100h        ; Next 4 lines are
8   mov    dword ptr [ecx+4],200h      ; Helper's inlined constructor
9   mov    dword ptr [ecx+8],400h
10  mov    dword ptr [ecx+0Ch],0F84030h
```

Beyond the code

Compiler optimizations may “optimize” security checks [7,8]

Example with overflow check:

```
unsigned int len;
...
if (ptr + len < ptr || ptr + len > max) return EINVAL;
```

- For the compiler, `ptr + len < ptr` can mean `len < 0`
- this is impossible (`len` is unsigned).

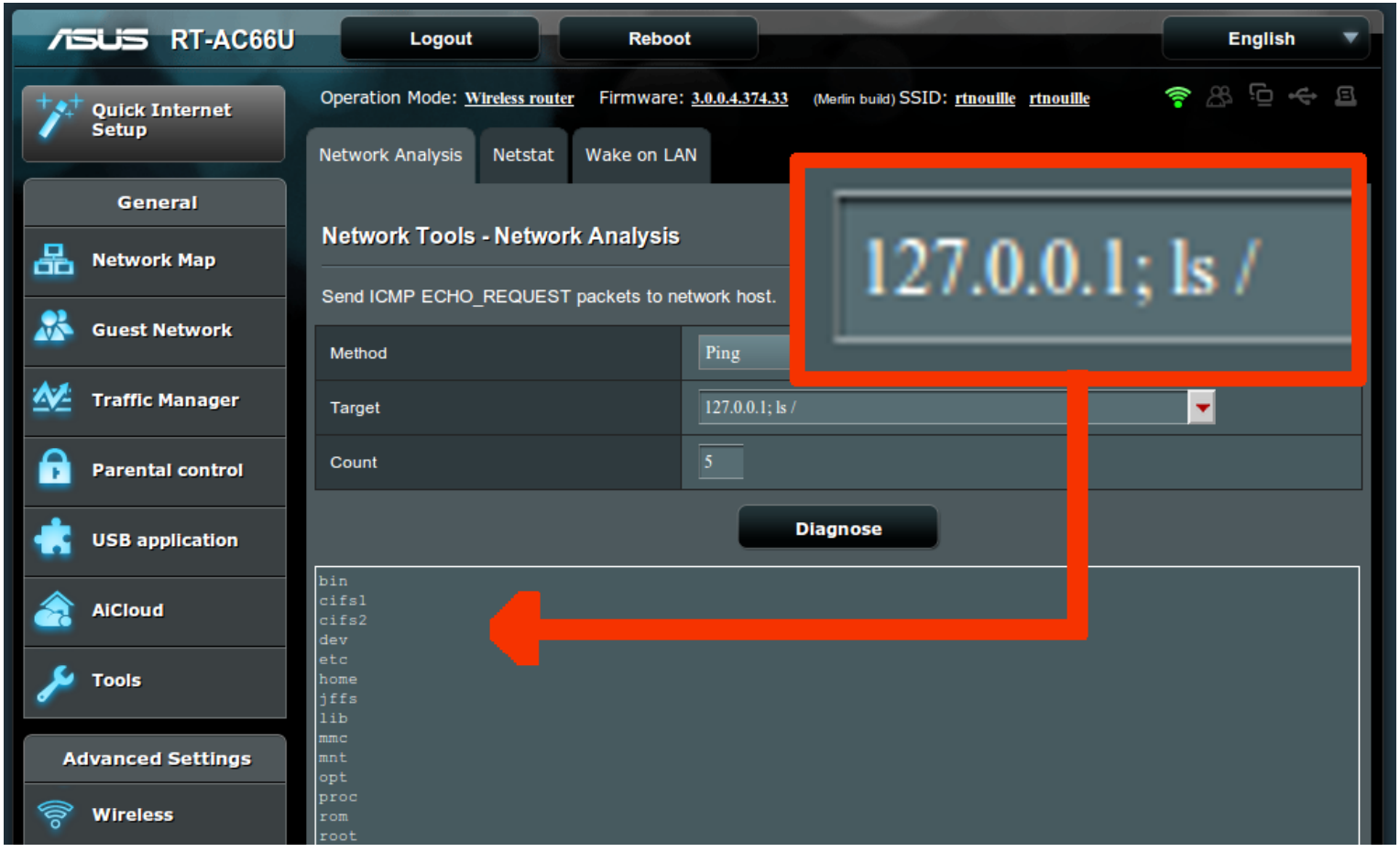
$\Longrightarrow$ the overflow check can be optimized out

- Could be rewritten `len > max-ptr`

[7] `http://www.kb.cert.org/vuls/id/162289`
[8] `http://bsidespgh.com/2014/media/speakercontent/DangerousOptimizationsBSides.pdf`

Good old injection

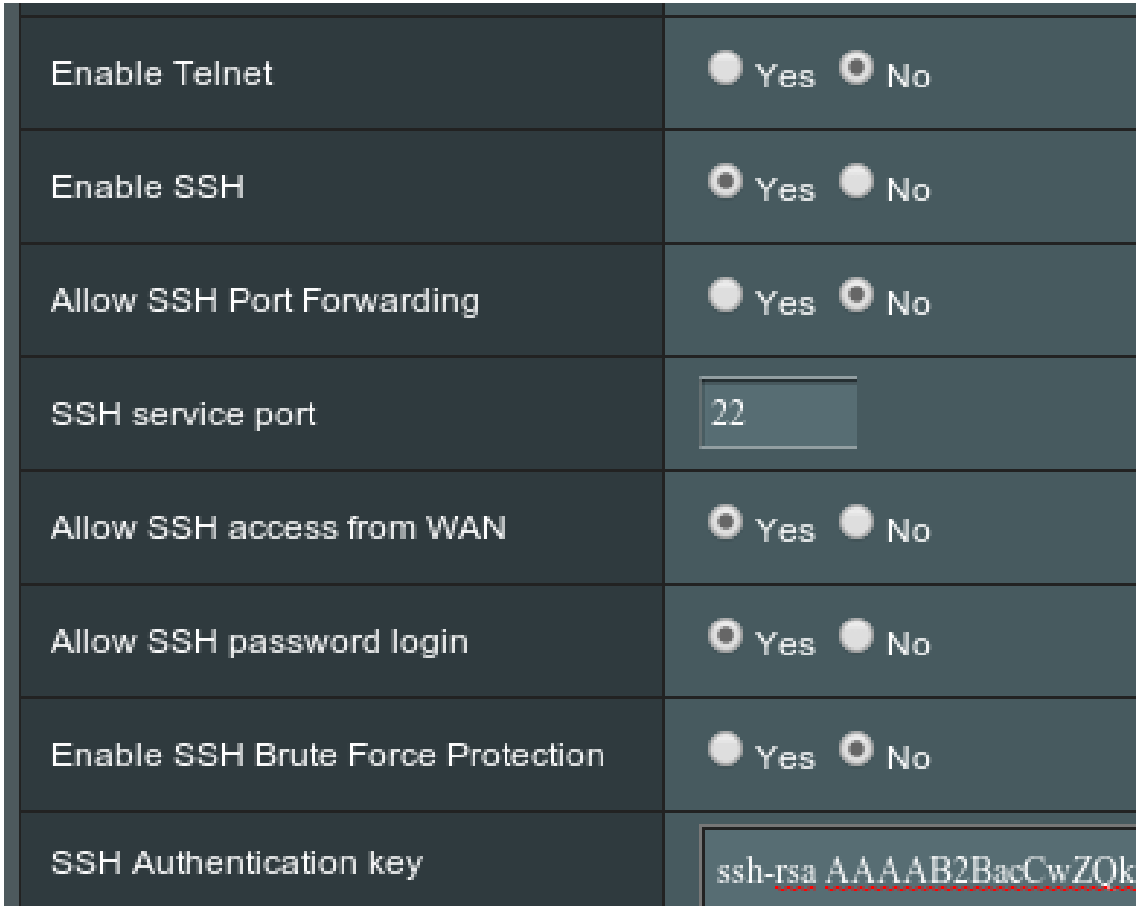
W00t! I just rooted my router!



Good old injection

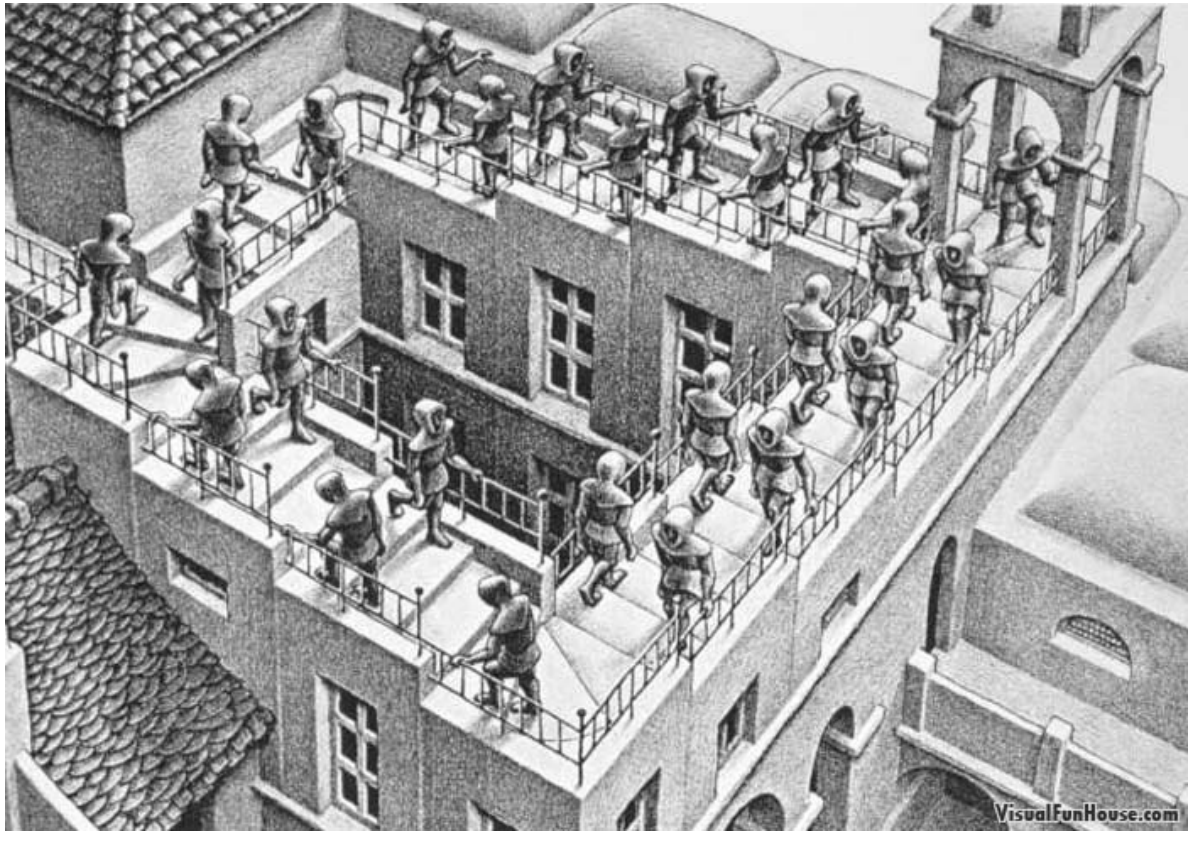
On another tab, not so far away

Oh! Actually I was already root.



Good old injection

Escalate privileges to ... where you already are



Whois stack buffer overflow (CVE-2003-0709)

The bug and the fix

The textbook case of buffer overflows

```
$ whois -g $(perl -e "print 'A'x2000")
Segmentation fault
```

```
- sprintf(p--, "-%c %s ", ch, optarg);
+ snprintf(p--, sizeof(fstring), "-%c %s ", ch, optarg);
```

Whois stack buffer overflow (CVE-2003-0709)

Impact

- non-privileged program ; not SUID

$\Longrightarrow$ escalate your privileges to ... where you already are ?

- what about all the websites proposing a whois service that actually ran whois through a CGI ?

$\Longrightarrow$ **escalate your privileges from anonymous web client to local shell**

Shellshock

Hard to analyze impact

- Bug: bash allows attackers to execute commands through specially crafted environment variables
- Impact: web servers using CGI scripts
- Impact: OpenSSH: users can bypass ForceCommand with `SSH_ORIGINAL_COMMAND`
- Impact: DHCP clients: some call bash scripts and transmit DHCP server parameters through environment variables
- ...

Debian/OpenSSL crypto-disaster

Very hard to analyze impact

- Bug: entropy for key generation limited to 15 bits
- Impact: SSL/TLS and X509 certificates
- Impact: ssh host and user keys
- Impact: Tor relays
- Impact: DH sessions keys can be recovered: PFS is broken. Impact is in the past!
- Impact: strong DSA keys can be recovered when used with a bad RNG! Impact is contagious!
- ...

Outline

1 Bugs!

2 Avoiding and Finding bugs

3 Bugs still happen

4 Why do bugs still happen ?!

5 Living with bugs

Best practices

- Software Configuration Management / Version Control
- Bug tracker
- Coding style

Software engineering

- Software architect
- Requirements
- V-Cycle, Agile methods, . . .
- Procedures

Assurance levels

- MISRA software guidelines
- ISO 26262
- DO-178b
- ...

Formal methods

- Model checking
- Abstract interpretation
- Theorem provers

Audits and tests

- Test campaigns
- Automatic tests (Find calls to dangerous functions like `system()`, `strcpy()`, `...`)
- Fuzzing

Certifications

- Common Criteria
- DO-178C (Software considerations in airborne systems and equipment certification)
- ...

Outline

1 Bugs!

2 Avoiding and Finding bugs

3 Bugs still happen

4 Why do bugs still happen ?!

5 Living with bugs

USS Yorktown

- 1996: used as a Smart Ship program test bed: 27 dual 200 MHz Pentium Pro
- 1997: crew member enters a zero into a database field

$\Longrightarrow$ division by zero

$\Longrightarrow$ crashes all computers

$\Longrightarrow$ propulsion system fails

$\Longrightarrow$ ship is dead in the water for 3h

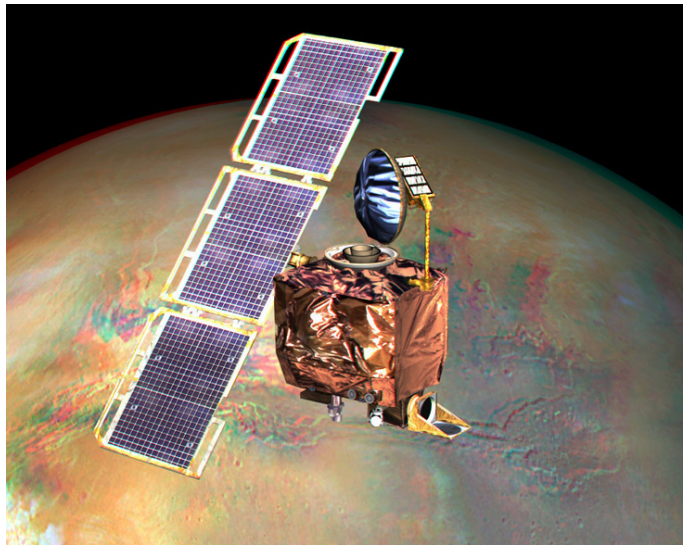
F22 raptor[9]

- First flight from Hawaii to Japan
- All system crashed when crossing latitude 180°
- Had to follow their tankers to go back home

[9] http://www.theregister.co.uk/2007/02/28/f22s_working_again/

Mars climate orbiter[10]

- one team used English units (inches, feet, etc.)
- another used metric units
- no need to say more



[10] http://www.jpl.nasa.gov/news/releases/99/mcoloss1.html

Patriot Missile[11]

- Time tracked by 0.1 increments
- 0.1 has no exact representation as a binary floating point
- Time tracking slowly drifted
- 0.3s drift in 100h
- 0.3s drift equals 600m at missile speed equals it can't follow its target
- workaround: reboot the system regularly

[11] https://en.wikipedia.org/wiki/MIM-104_Patriot#Failure_at_Dhahran

787 Dreamliner[13]

A Model 787 airplane that has been powered continuously for 248 days[12] can lose all alternating current electrical power due to the generator control units simultaneously going into failsafe mode,

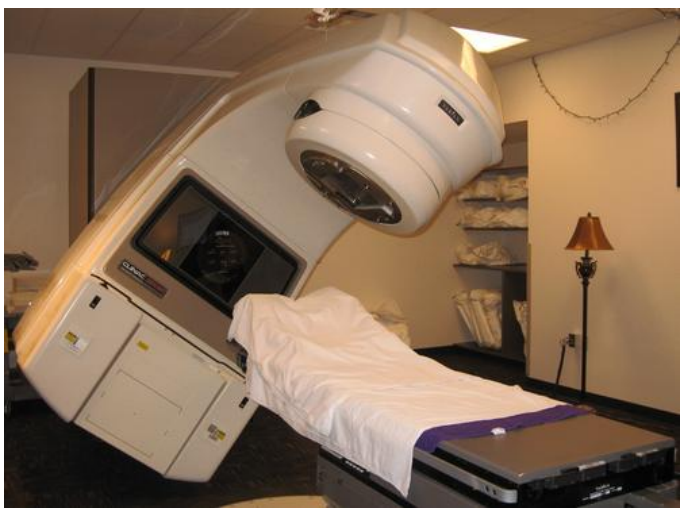
- 248 days $= 2^{31}$ 100^{th} of a second
- coincidence ?

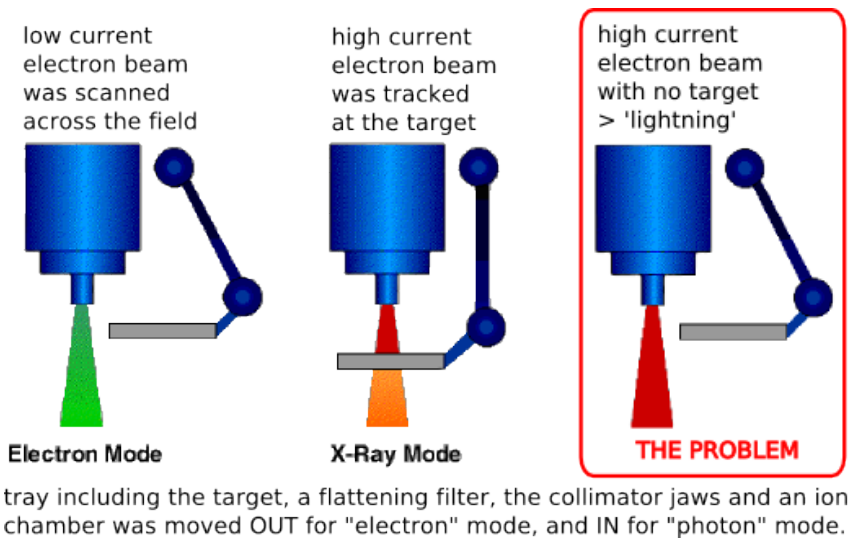
[12] this should not happen in normal operational conditions

[13] `http://www.engadget.com/2015/05/01/boeing-787-dreamliner-software-bug/`

Therac 25[14,15]

- Radiotherapy machine used in 80's
- VT-100 terminal connected to PDP-11 computer driving the device
- Two modes:
  - Direct low energy electron beam
  - X-Ray created from high energy electron beam hitting a target





tray including the target, a flattening filter, the collimator jaws and an ion chamber was moved OUT for "electron" mode, and IN for "photon" mode.

[14] https://en.wikipedia.org/wiki/Therac-25

[15] http://web.mit.edu/6.033/www/papers/therac.pdf

Therac 25

How this was possible

Big engineering failure

- no hardware interlocks to prevent high energy mode without target (previous models had it)
- open-loop controller: the software could not check the device was working correctly
- a flag was set and reset by incrementing and decrementing it. Sometimes overflow occurred.
- when hitting X (X-Ray), ↑ then E (change X-Ray to Electron beam), then ↵ then B (beam on) in less than 8s
- system displayed `MALFUNCTION 54` ; no explanation in the manual ; operator press P to proceed anyway
- vendor always denied that overdose could be possible

Toyota Unintended Acceleration[17]

- Some critical variables are not protected from corruption
- No hardware protection against bit flips
- Buffer Overflow, Invalid Pointer Dereference and Arithmetic, Race Conditions, Unsafe Casting, Stack Overflow (bug bingo!)
- Cyclomatic Complexity[16] over 50 (untestable) for 67 functions. Over 100 for the throttle angle function.
- Used Recursion (dangerous with fixed size stack) ; failed the worst-case stack depth analysis
- Watchdog only monitored 1 task out of 24
- and too many more to fit here!

[16] measure of the complexity of the control flow graph
[17] `http://www.sddt.com/files/BARR-SLIDES.pdf`

Outline

1 Bugs!

2 Avoiding and Finding bugs

3 Bugs still happen

4 Why do bugs still happen ?!

5 Living with bugs

Best practices

- best practices are not followed
- tools are not used

Formal methods

Formal methods did not prevent them because

- They were invented after most of those event happen precisely to prevent them from happening again
- They were not used (time/money constraints, incompetence)
- They cannot be applied yet to most our non-critical software (Openssl, Javascript code, . . .)
- They only find what they have been made to look for

Complexity

- system are more and more complex
- we are not smarter!

Human condition

- tiredness, mood, hangover, . . .
- working memory is volatile
  - lasts at most 20s
  - stands no interruption
- working memory can hold only 7 ± 2 things

High cognitive load

Low cognitive load

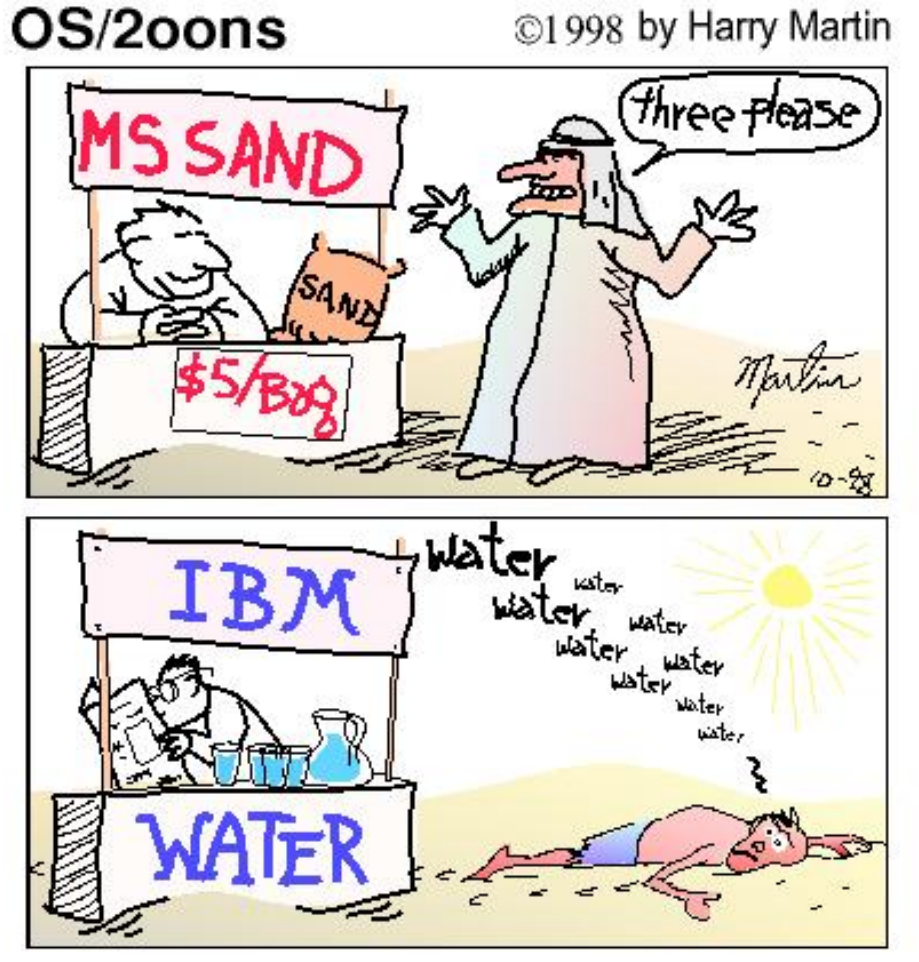
Low cognitive load too

Communication issues

- same units ?
- ambiguous API ?

Natural selection vs Marketing

Illustrated by Windows winning over OS2



Marketing mysteries of the Universe.

End Users

They make mistakes. They are unpredictable.

Outline

1 Bugs!

2 Avoiding and Finding bugs

3 Bugs still happen

4 Why do bugs still happen ?!

5 Living with bugs

Keep it simple

KISS: Keep It Simple, Stupid.

Hardening, Compartmentalization

- Least privilege
- Privilege separation
- SE Linux, App Armor
- PaX, GrSecurity
- Sandboxes

Giant bags of mostly water

Very interesting parallel with car safety[19,20]

In the 60's:

- whistle blowers:

  "Vehicle interiors are so poorly constructed from a safety standpoint that it is surprising that anyone escapes from an automobile accident without serious injury."

  – Journal of the American Medical Association, 1955

  Unsafe at any speed[18]

- engineers:
  - Cars are safe – they do not explode, catch fire, ...
  - Accidents are due to bad drivers
  - Educating drivers will solve the problem

Sounds familiar ?

[18] https://en.wikipedia.org/wiki/Unsafe_at_Any_Speed
[19] http://kernsec.org/files/lss2015/giant-bags-of-mostly-water.pdf
[20] https://www.youtube.com/watch?v=C_r5UJrxcck

Giant bags of mostly water

Nowadays cars

Mindset changed:

- 3 point seat belts ; pre-tensioners
- airbags everywhere
- ABS
- Electronic Stability Control
- Head Injury Protection
- life module
- collision sensors
- independent mandatory crash tests and public rating
- . . .

Shame

- Having lice was synonym of poverty and bad hygiene

$\Longrightarrow$ people were ashamed to have them

$\Longrightarrow$ did not tell anyone

$\Longrightarrow$ other children could be infested without noticing (only eggs for instance)

$\Longrightarrow$ infestation would come back over and over

- Mindset has changed
- When one child has some, all the classroom is informed

$\Longrightarrow$ children are checked and cleaned during the same time period.

Sounds familiar ?

ICFP'99 programming contest: Optimizing non-player characters

http://www.cs.tufts.edu/~nr/icfp/problem.html

```
((0 1 2 3 4 9 20 (IF (AND (EQUALS (VAR "where") 1) (EQUALS (VAR "v
         (EQUALS (VAR "state") 0)) (DECISION 0
              "^A parrot perches on a branch high up in the elm tr
    ((ELSEIF (AND (EQUALS (VAR "verb") 0) (EQUALS (VAR "state") 0)
              (DECISION 0
                        "^A parrot sits half-hidden among the bran
     (ELSEIF (AND (EQUALS (VAR "verb") 6) (EQUALS (VAR "state") 1)
              (DECISION 3
                        "Your throw goes wild, and you barely brus
      (ELSEIF (AND (EQUALS (VAR "state") 1)) (DECISION _ ""))
      (ELSEIF (AND (EQUALS (VAR "verb") 10) (EQUALS (VAR "state")
              (DECISION _ "The parrot takes no notice of you."))
      (ELSEIF (AND (EQUALS (VAR "verb") 10) (EQUALS (VAR "state")
              (DECISION _ "The parrot takes no notice of you."))
      (ELSEIF (AND (EQUALS (VAR "verb") 10) (EQUALS (VAR "state")
              (DECISION _ "The parrot takes no notice of you."))
[...]
```

ICFP programming contest: Optimizing characters

- Character files are compiled into a program
- Grammar, semantics, time and size of each instruction are given
- You must create a program that optimize a character file in size and time
- Your program must run in less than 30 minutes
- You have 72h to write your program

ICFP programming contest: Optimizing characters

- Complex problem
- Limited time

$\Longrightarrow$ There will be ~~blood~~ bugs!

ICFP programming contest: Optimizing characters

- The input is a valid output
- Better give a non-optimized valid answer than a wrong answer or no answer at all
- Easy to compare an answer with the input by evaluating it on several points

Winning team solution[21]

- Used a supervisor
- Initialize a variable with the input
- Run several optimizers
- Each time an answer is proposed, it is tested
- if correct and better than the current answer, replace it
- at 29m30s, output the current best answer

[21] `http://caml.inria.fr/pub/old_caml_site/icfp99-contest/`

The Chaos Monkey[22,23]

- In the cloud, resilient architectures should handle the crash of a machine
- The Chaos Monkey runs in the Amazon Web Services (AWS)
- It randomly terminate instances (during working hours)

the best defense against major unexpected failures is to fail often

Netflix

[22] `http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html`
[23] `https://github.com/Netflix/SimianArmy`

Conclusion

- Defense in depth
- Everything can fail
- Make things that can work in degraded mode
- Use supervisors, watchdogs
- Think one move ahead

Conclusion

Failure is not an option*

*option: something that you can avoid