MY FAVORITE THINGS

Sergey Bratus

TRADITIONAL

- Raindrops on roses,
- Whiskers on kittens,
- Bright copper kettles,
- Warm woolen mittens,
- …

H.P. LOVECRAFT'S

- Shoggoths that glibber
- and ghouls that go meeping,
- Eldritch dark ichor,
- and the dead never sleeping;
- Night-gaunts that flap with their blasphemous wings,
- these are a few of my favorite things.

http://transform.to/~mjc42/tut/library/humour.html

H.P. LOVECRAFT'S

- Shoggoths that glibber
- and ghouls that go meeping,
- Eldritch dark ichor,
- and the dead never sleeping;
- Night-gaunts that flap with their blasphemous wings,
- these are a few of my favorite things.

http://transform.to/~mjc42/tut/library/humour.html

MY FAVORITE THINGS

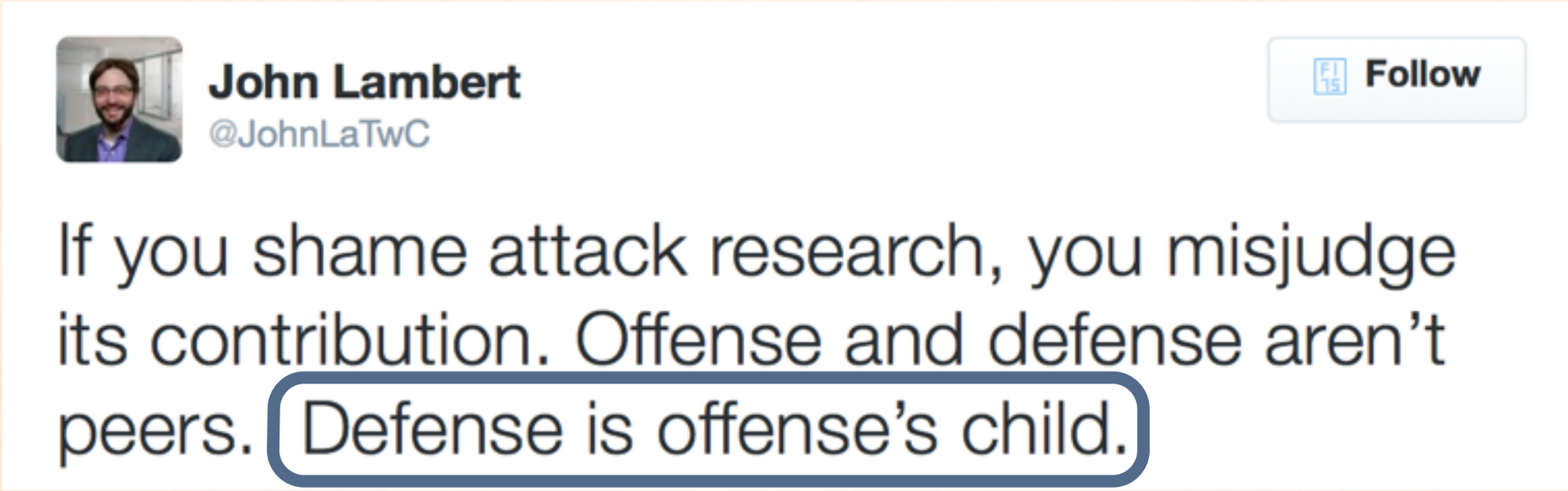
- The halting problem & friends
  - “I’d rather write programs ~~to write~~ to run on programs than write programs”
- Parser differentials
  - in every OSI model layer!
  - "even more undecidability!"

DIFF KEYNOTE.{1ST,2ND}

- **Hard** vs (provably) **Impossible**
  - "Hard" will get figured out, impossible will keep failing
  - Hard: flight. Impossible: perpetual motion
- Not all **complexity** is created equal
  - Landscape has **cliffs & sheer drops into the abyss**
  - We must know & avoid them. All other kinds of engineers do!

DIFF KEYNOTE.{1ST,2ND}

- Offense creates security science
  - Exploits are proofs. In traditional sciences, "zero-day" is simply called "new result" (a.k.a. "worth publishing")
  - "A theory of security comes from a **theory of insecurity**"



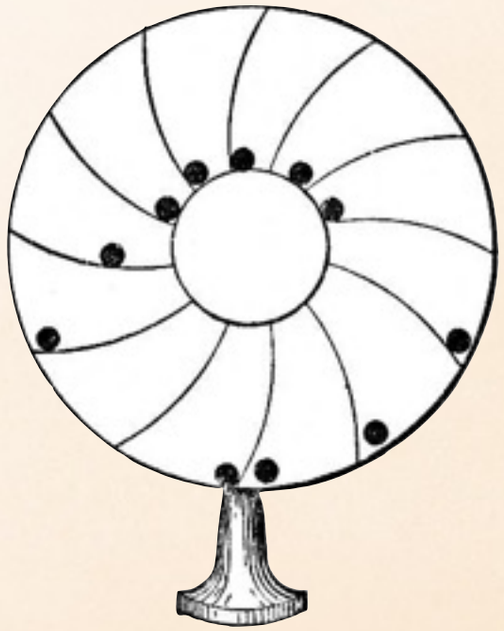
“THE DARK SIDE”

How you learned about software

How it actually works

IMPOSSIBILITY STRIKES BACK

- "Natural law": you can't stop nature from doing **this** no matter **how hard** you try
  - Perpetual motion 1st kind (free work without energy input)
  - Lossless energy transformations (2nd kind, no energy leaks)
  - Speed of light, Heisenberg's uncertainty, ...



WHAT'S YOUR IMPOSSIBILITY?

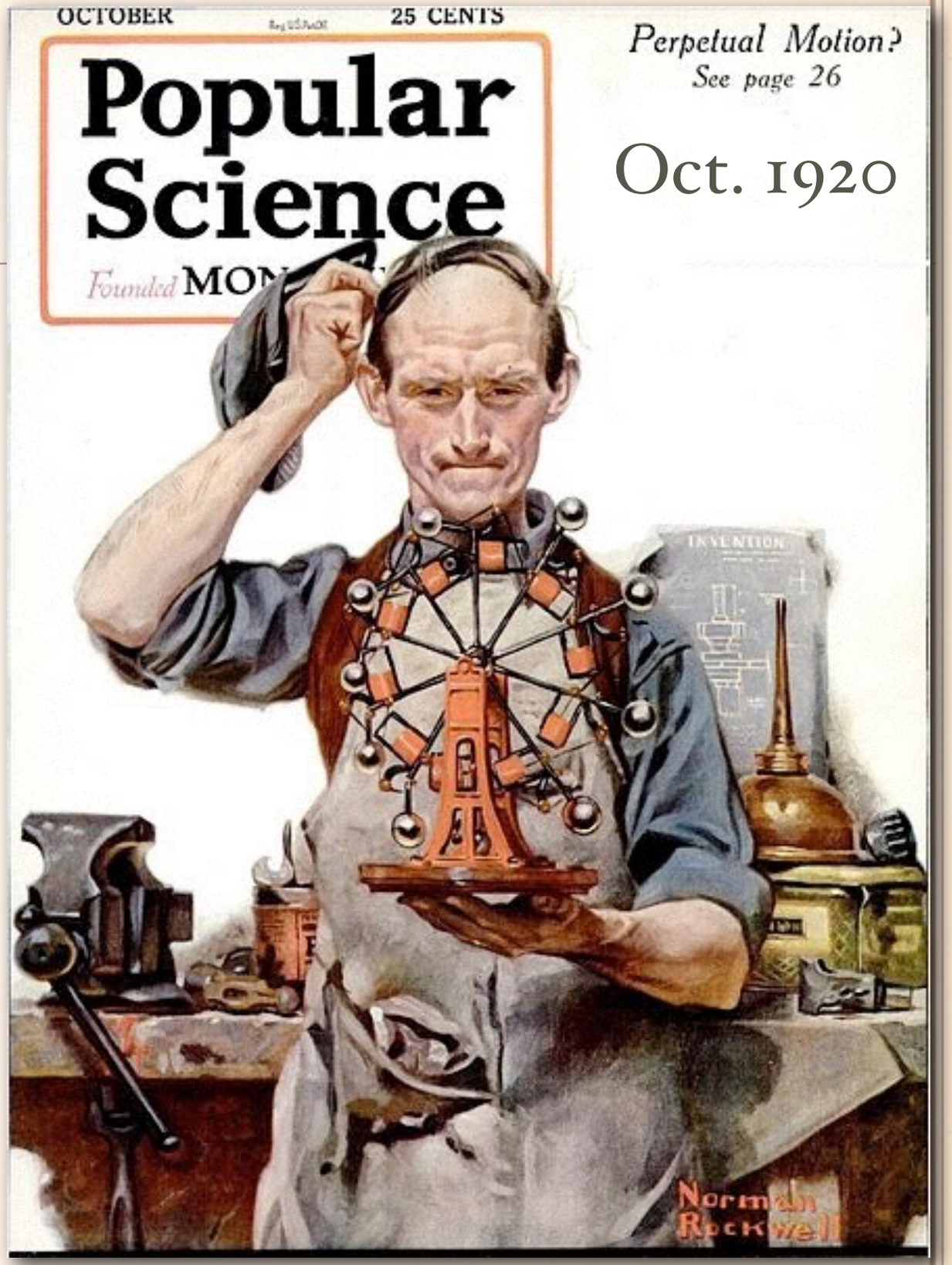
- Physical world **engineering** is **defined** by physical **impossibilities**
- Impossibility doesn't mean we are doomed, it just means an engineer must:
  - Know the limiting laws
  - **Never** base designs on hopes of cheating them
    - unless, of course, your intent is <u>sabotage</u>.

ASK AN ENGINEER

- What's your **impossibility**? What's wrong to attempt? What your design should never depend on solving?
  - *Mechanical:* conservation laws, ...
  - *Thermal:* thermodynamics laws, ...
  - *Computer:* energy dissipation, latency < speed of light, quantum effects, ...
  - *Software:* **???** (crypto? maybe...)

Oh ye seekers after perpetual motion, how many vain chimeras have you pursued? Go and take your place with the alchemists.

da Vinci, 1494

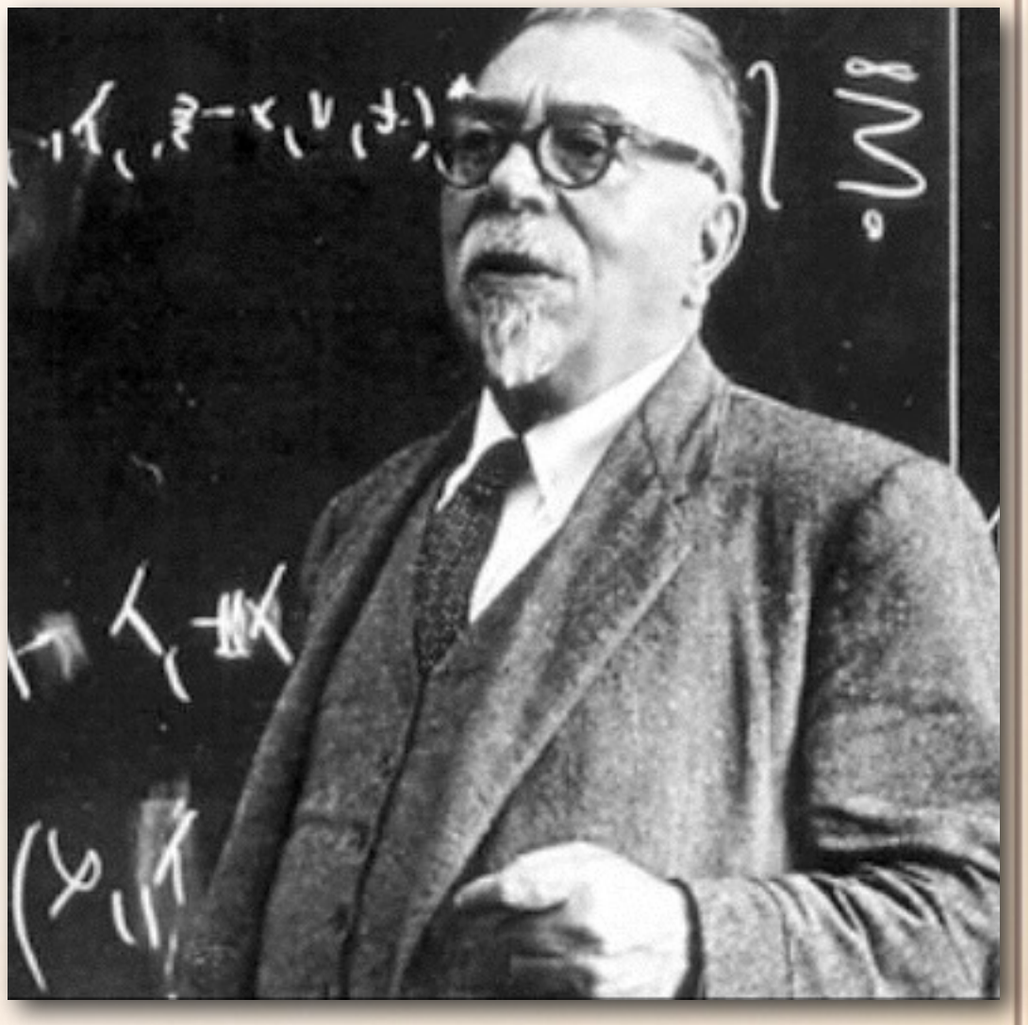


Oct. 1920

CYBERNETICS?

"One of the chief duties of the mathematician in acting as an adviser to scientists is to **discourage** them from expecting too much from mathematics."

-- Norbert Wiener, **1964**



COMPUTERS CAN IMPROVE EVERYTHING!

"Since **symbols** can be written and moved about with **negligible** expenditure of **energy**, it is tempting to leap to the conclusion that **anything is possible** in the symbolic realm. This is the lesson of *computability theory* (viz., solvable problems vs. unsolvable problems), and also the lesson of *complexity theory* (viz., solvable problems vs. feasibly solvable problems): **physics does not suddenly break down at this level of human activity**. It is no more feasible to construct *symbolic* structures without using energy than it is possible to construct *material* structures for free."

Richard A. DeMillo, Richard J. Lipton, and Alan J. Perlis, **1979**
'Social Processes and Proofs of Theorems and Programs'; Yale tr82

CYBERCYBER!

"One of the chief duties of the ~~mathematician~~ *computer scientist* in acting as an adviser to ~~scientists~~ *everyone* is to discourage them from expecting too much from ~~mathematics~~ *computers*"

-- stolen from Norbert Wiener, 2013

CYBERCYBER!

"One of the chief duties of the ~~mathematician~~ ~~*computer scientist*~~ **hacker** in acting as an adviser to ~~scientists~~ *everyone* is to discourage them from expecting too much from ~~mathematics~~ *computers*"

-- stolen from Norbert Wiener, 2013

INPUT IS "CYBER KRYPTONITE!"

- Programs are bad at analyzing programs
- All inputs are programs

- Programs are bad at analyzing **inputs**
  - we must know & avoid impossibilities

SCOOPING THE LOOP SNOOPER

A proof that the Halting Problem is undecidable

Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

HALTING PROBLEM

- ***"I heard you had a program for analyzing programs, so I put a program that analyzes programs into a program for you to analyze"***

- "Let **h**(x,i) = 1 if program x halts on input i, 0 otherwise"

- "for any totally computable function **f**(x,y), **h**(*g*,*g*) **!=** **f**(*g*,*g*) for the program *g* that implements

f(*g*,*g*)=0 => g(*g*) = 0 => **h**(*g*,*g*)=1

f(*g*,*g*)=1 => g(*g*) loops => **h**(*g*,*g*)=0

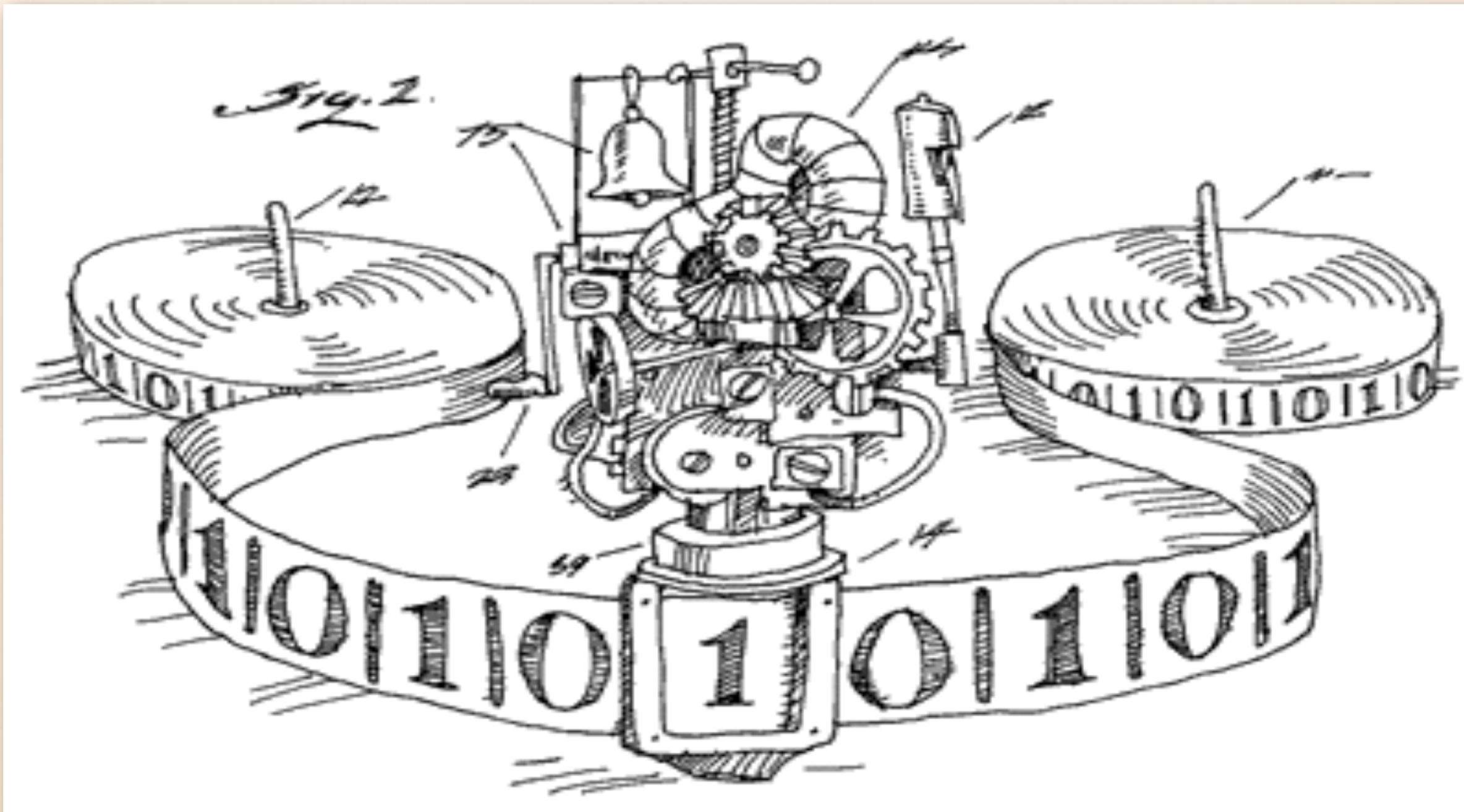
```
procedure compute_g(i):
    if f(i,i) == 0 then
        return 0
    else
        loop forever
```

HAVE YOU HEARD THIS BEFORE?

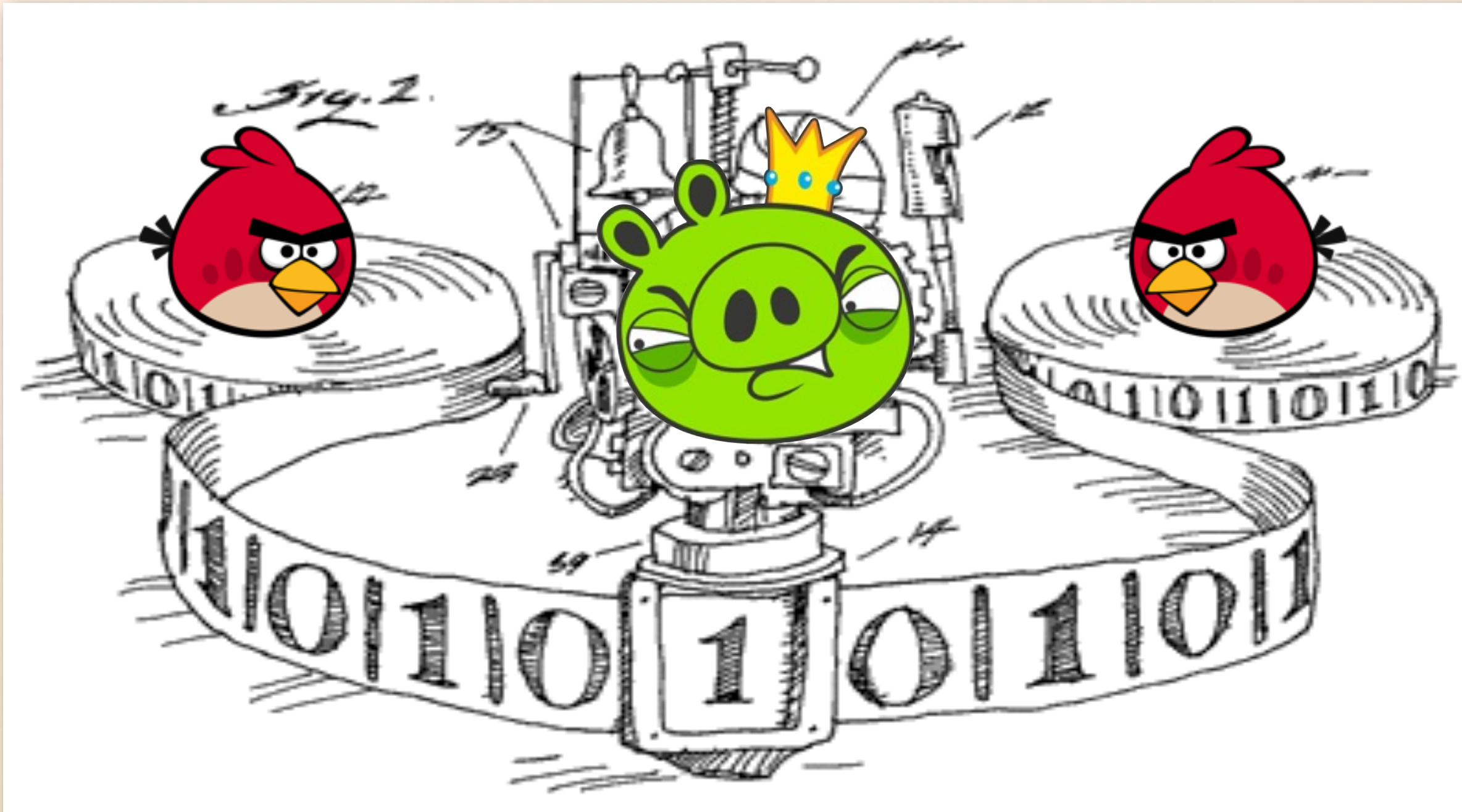
Bertrand Russell loves you and wants you to be happy

A ~~barber~~ hacker can only ~~shave~~ hack those who don't hack themselves. Can the hacker hack himself?

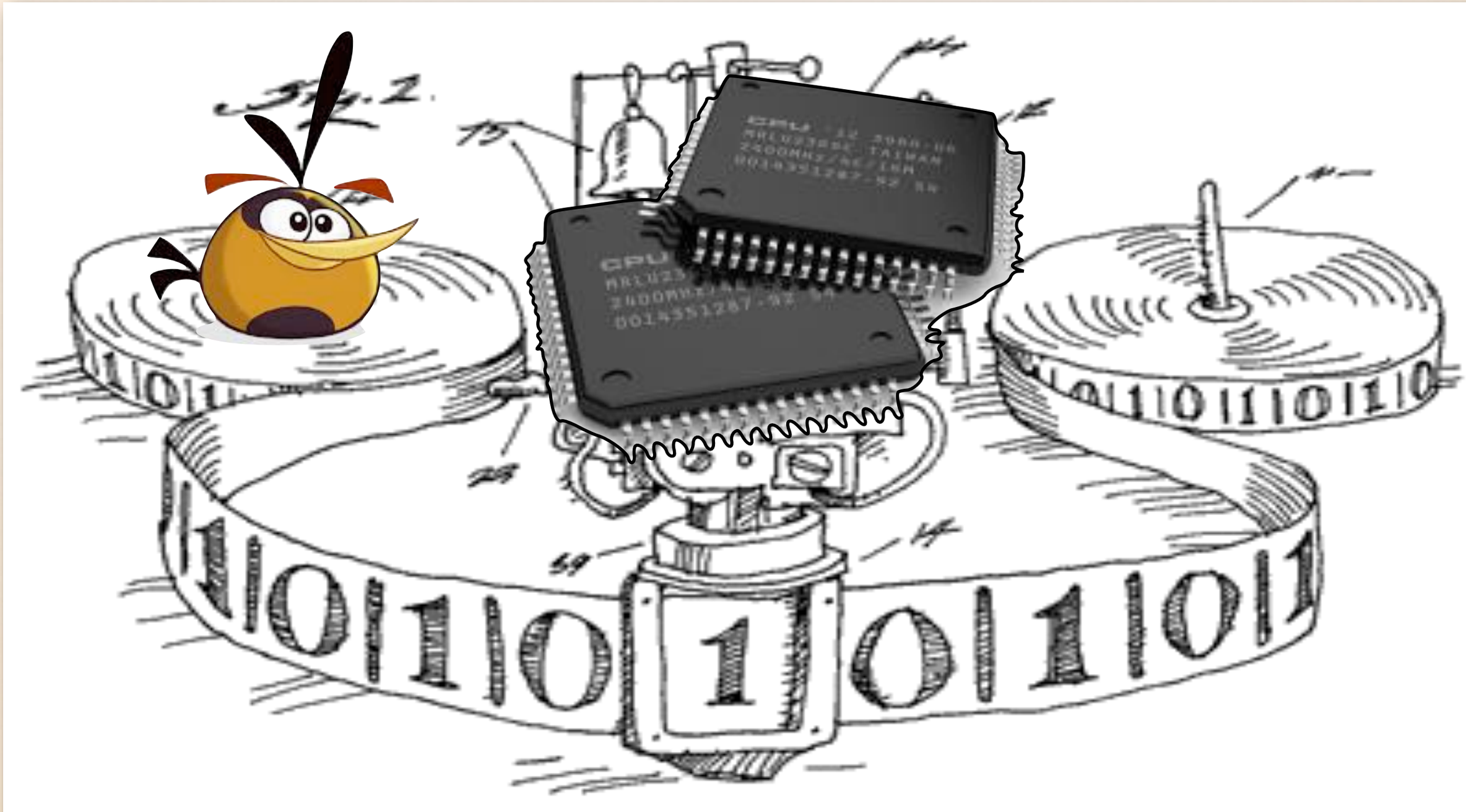
INPUTS VS PROGRAMS



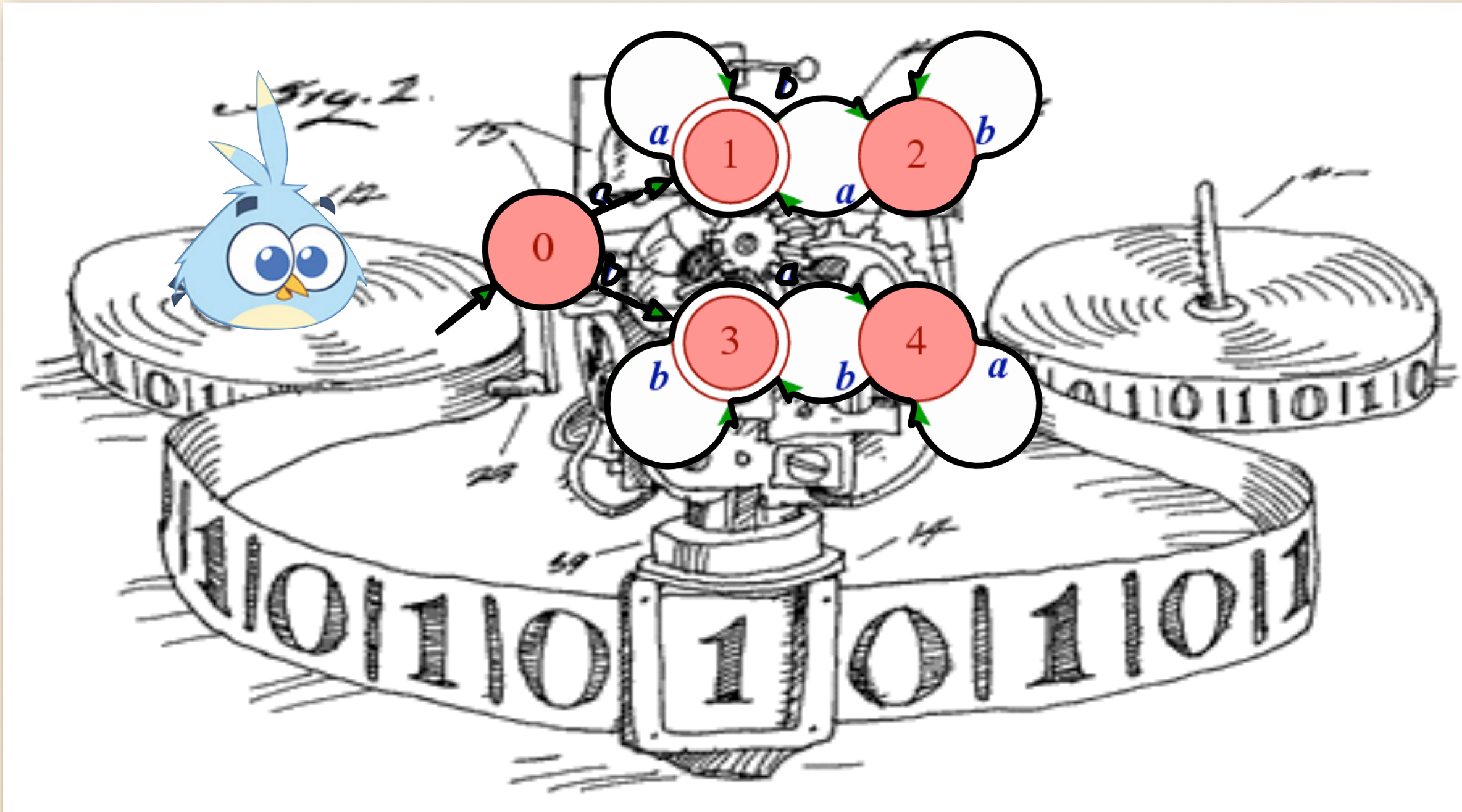
INPUTS VS PROGRAMS



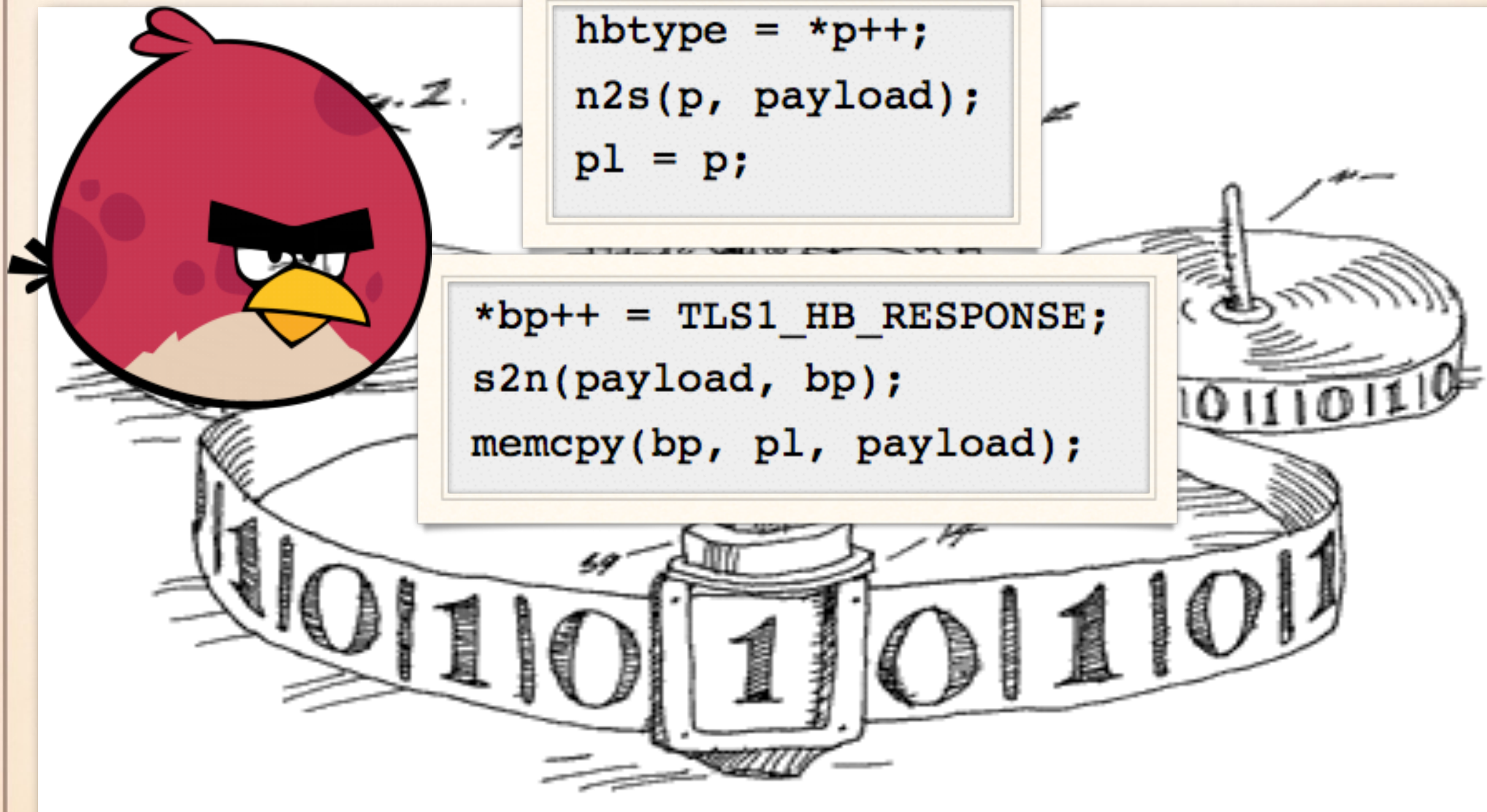
INPUTS VS PROGRAMS



INPUTS VS PROGRAMS



INPUTS VS PROGRAMS



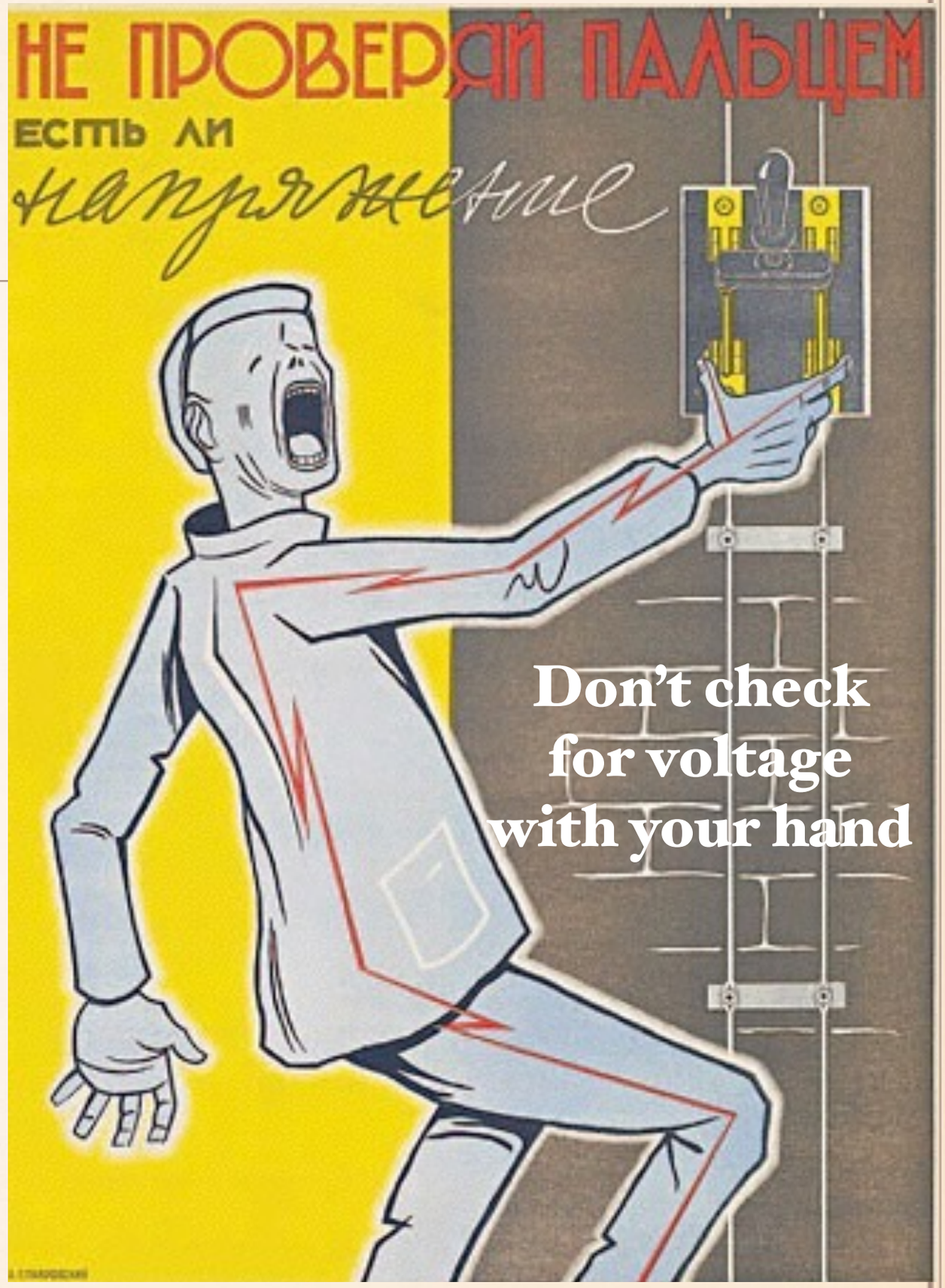
```
hbtype = *p++;
n2s(p, payload);
pl = p;
```

```
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
```

HINDSIGHT IS 20/20, RIGHT?

HINDSIGHT IS 20/20, RIGHT?

- **Workplace safety rules** are **hindsight**, too
  - "written in blood"
- Such hindsight is long overdue in software!




КИРПИЧ
УКЛАДЫВАЙ
ПРАВИЛЬНО
25
ЗДОВ


КИРПИЧ
УКЛАДЫВАЙ
ПРАВИЛЬНО
25
РЭФОБ


НАВЕРХУ
РАБОТАЮТ
НЕ
СТОЙ
ПОД МАЧТОЙ


КИРПИЧ
НЕ ВЕДИ
РАБОТУ
ПОДКОПОМ


НАВЕРХУ
РАБОТАЮТ
НЕ
СТОЙ
ПОД МАЧТОЙ

"A BRIGHT LINE FOR INPUTS"

Checks

Input validation

Recognition

malloc( )

memcpy( )

+, - , *, /

THE COMMON FAILURE PATTERN

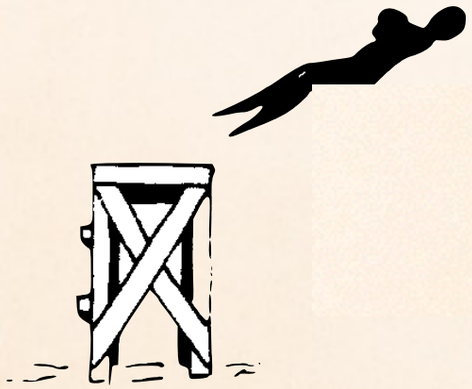
"Sanity Checks"

+, - , *, /

malloc( )

"Input sanitization"

memcpy( )



THE COMMON FAILURE PATTERN

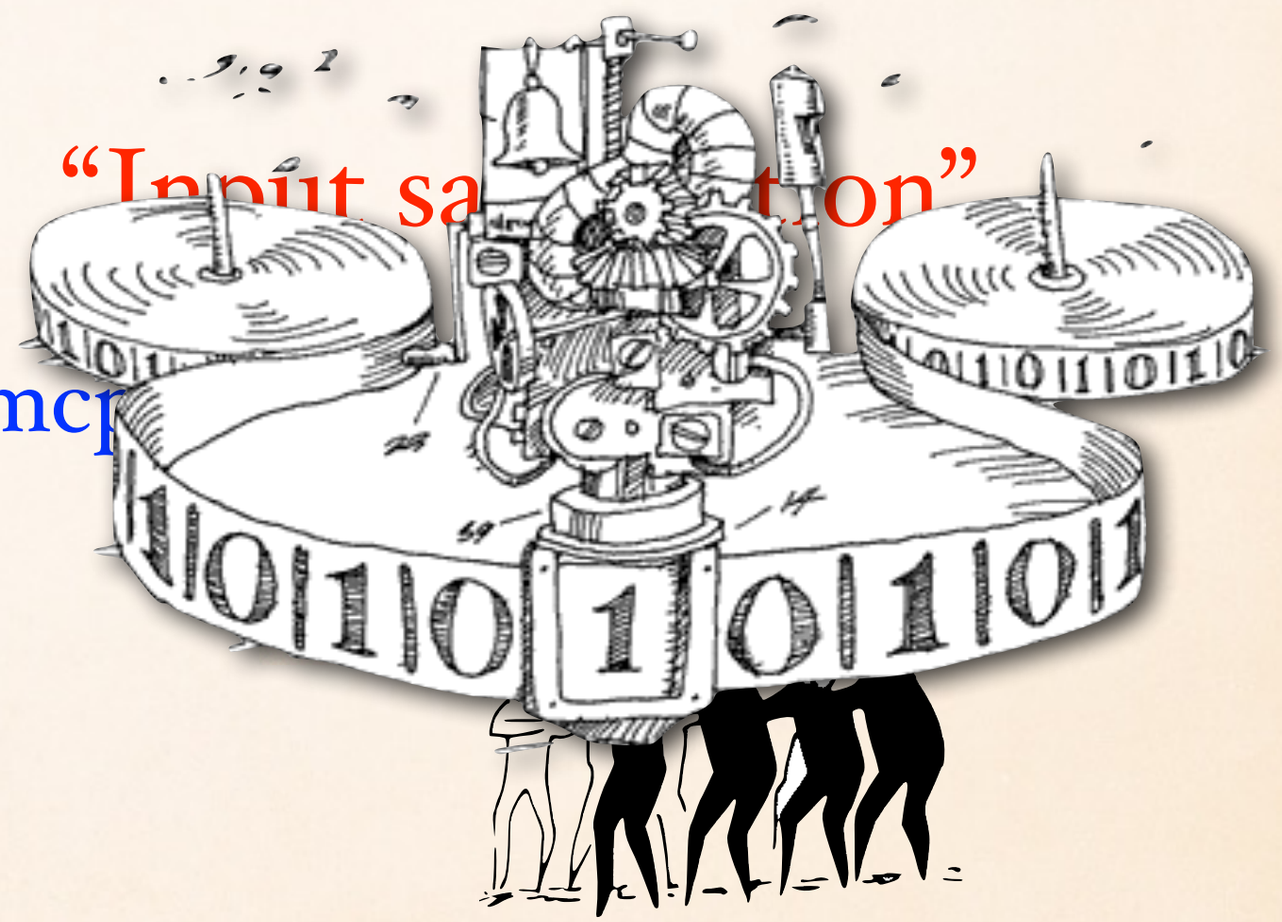
"Sanity Checks"

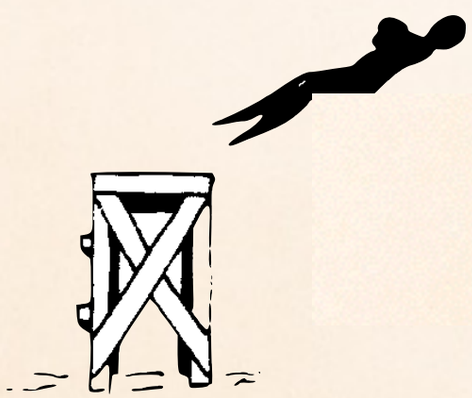
+, - , *, /

malloc()

"Input sa t on"

memcp





HEARTBLEED IS A PARSER BUG!

Heartbeat sent to victim

SSLv3 record:

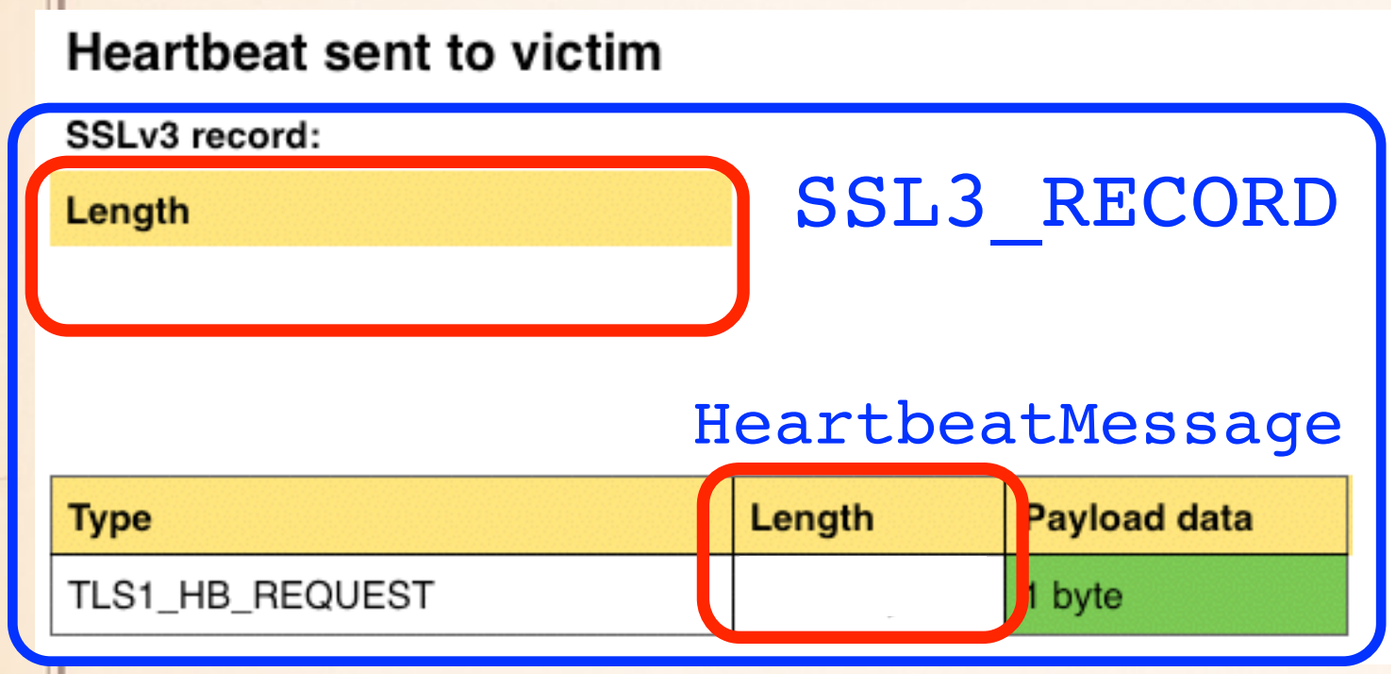
Length

SSL3_RECORD

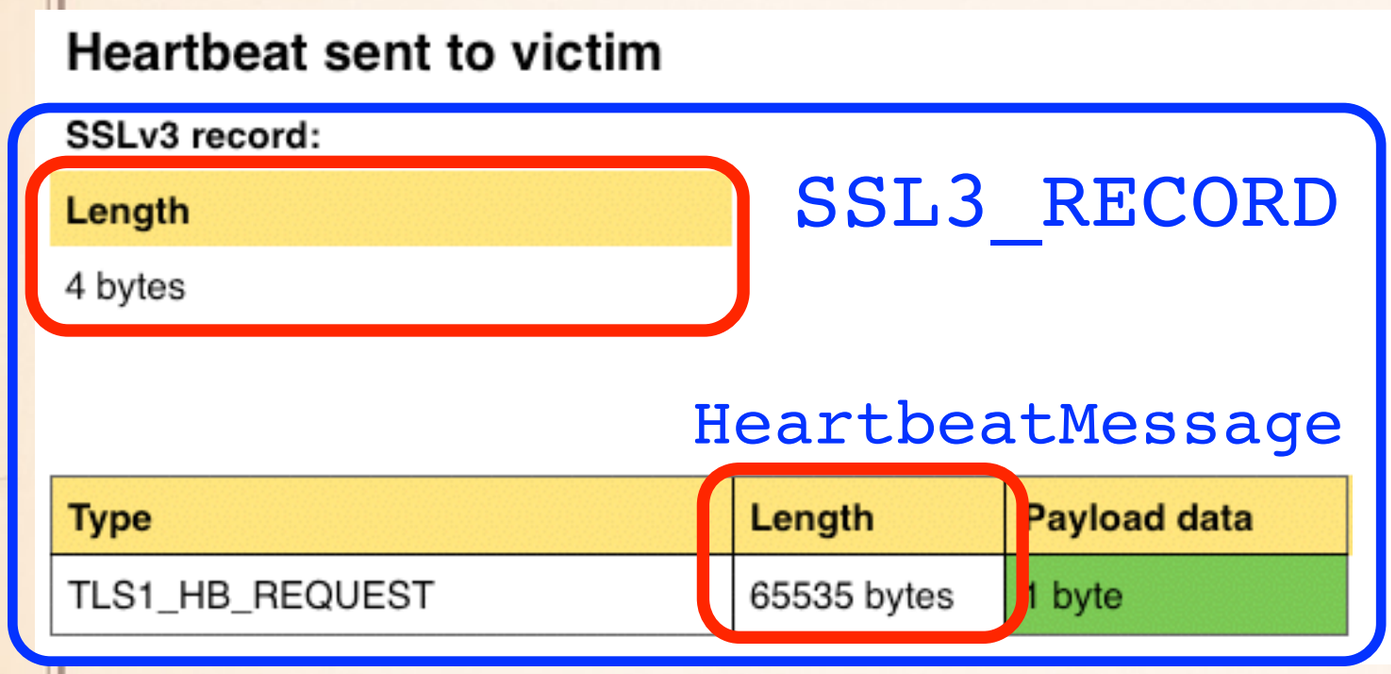
HeartbeatMessage

Type	**Length**	**Payload data**
TLS1_HB_REQUEST		1 byte

HEARTBLEED IS A PARSER BUG!



HEARTBLEED IS A PARSER BUG!



HEARTBLEED IS A PARSER BUG!

Heartbeat sent to victim

SSLv3 record:

Length
4 bytes

SSL3_RECORD

HeartbeatMessage

Type	Length	Payload data
TLS1_HB_REQUEST	65535 bytes	1 byte

Victim's response

SSLv3 record:

Length
65538 bytes

HeartbeatMessage:

Type	Length	Payload data	
TLS1_HB_RESPONSE	65535 bytes	65535 bytes	

HEARTBLEED IS A PARSER BUG!

Heartbeat sent to victim

SSLv3 record:

Length

4 bytes

Must agree, never checked

HeartbeatMessage:

Type	Length	Payload data
TLS1_HB_REQUEST	65535 bytes	1 byte

```
hbtype = *p++;
n2s(p, payload);
pl = p;
```

Victim's response

SSLv3 record:

Length

65538 bytes

```
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
```

HeartbeatMessage:

Type	Length	Payload data	
TLS1_HB_RESPONSE	65535 bytes	65535 bytes	

```
-       /* Read type and payload length first */
-       hbtype = *p++;
-       n2s(p, payload);
-       pl = p;

        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        s, s->msg_callback_arg);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
+               unsigned int write_length = 1 /* heartbeat type */ +
+                                           2 /* heartbeat length */ +
+                                           payload + padding;
                int r;
```

```
-               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
+               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, write_length);

                if (r >= 0 && s->msg_callback)
                        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
-                               buffer, 3 + payload + padding,
+                               buffer, write_length,
                                s, s->msg_callback_arg);
```

```
-       /* Read type and payload length first */
-       hbtype = *p++;
-       n2s(p, payload);
-       pl = p;

        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        s, s->msg_callback_arg);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
+               unsigned int write_length = 1 /* heartbeat type */ +
+                                           2 /* heartbeat length */ +
+                                           payload + padding;
                int r;
```

```
-               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
+               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, write_length);

                if (r >= 0 && s->msg_callback)
                        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
-                               buffer, 3 + payload + padding,
+                               buffer, write_length,
                                s, s->msg_callback_arg);
```

```
-       /* Read ty
-       hbtype =
-       n2s(p, pay
-       pl = p;

        if (s->ms
                s-



+       /* Read ty
+       if (1 + 2
+               re
+       hbtype =
+       n2s(p, pay
+       if (1 + 2
+               re
+       pl = p;
+
        if (hbtype
                {
                u
+               u
+
+
                i


-               r
+               r

                if


-
+
```

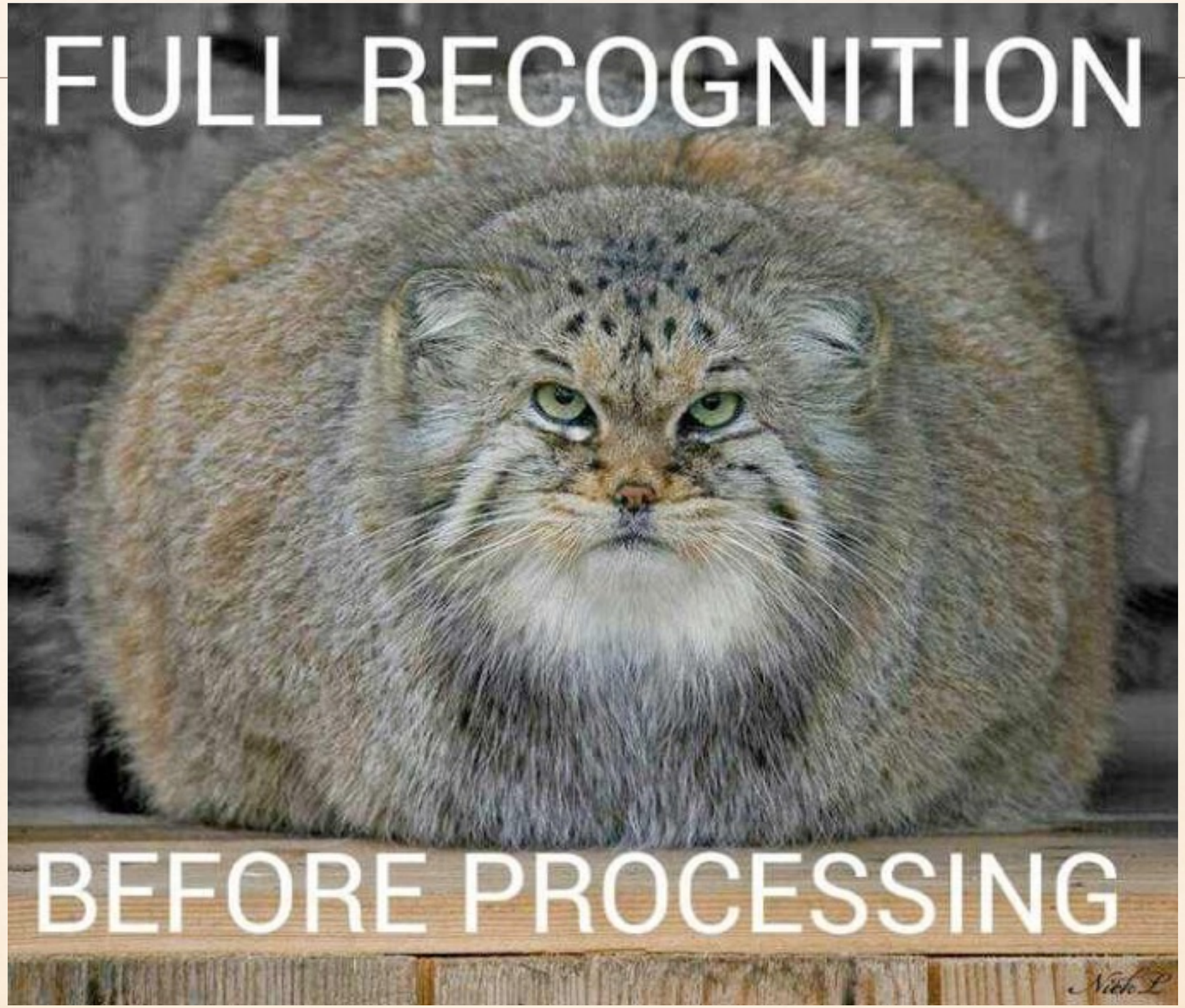
CH


```
-               oad + padding);
+               ngth);
```

Your input is a language;
treat it as such:
write a grammar spec!

PARSER CODE SHOULD
READ LIKE THE
GRAMMAR

FULL RECOGNITION



FULL RECOGNITION

```
/*

MANUL THE LANGSEC CAT SAYS:

      _ _
 ◄/       \►                ___ _   _ _   _
 ≻== ò ▾ ó ==≺             | __| | | | |  | |
   \   △  /   __           | _|| |_| | |__| |__
  /        \  |  \         |_|  \___/|____|____|
 /          \ |  |                                  before processing
| |         | ||  |
 \ |  ||  |  /  /            ___ ___ ___ ___   ___ _  _ ___ _____ ___ ___  _  _
  \_oo__oo_/___/            | _ \ __/ __/ _ \ / __| \| |_ _|_   _|_ _/ _ \| \| |
                            |   / _| (_| (_) | (_ | .` || |  | |  | | (_) | .` |
                            |_|_\___\___\___/ \___|_|\_|___| |_| |___\___/|_|\_|

*/
```

utf-8 manul by

Melissa Μέλισσα
@0xabad1dea
#126,030,998

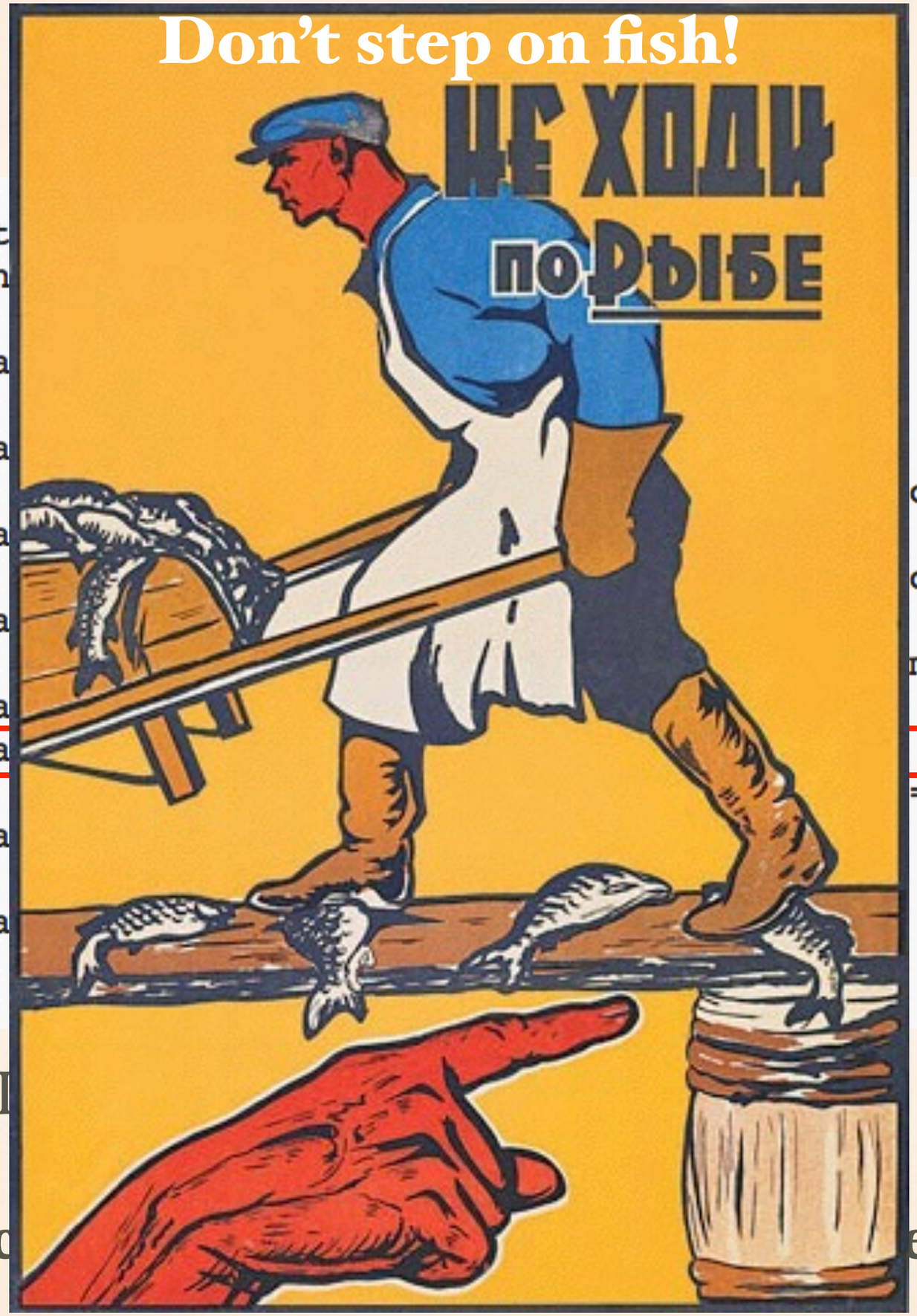
"GOTO FAIL"

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
. . .
```

- Apple's SSL state machine, **hand-coded**
- **State machine done wrong: code must be generated!**

Don’t step on fish!

```
. . .
hashOut.dat
hashOut.len
if ((err =
    goto fa
if ((err =
    goto fa
if ((err =                                  om)) != 0)
    goto fa
if ((err =                                  om)) != 0)
    goto fa
if ((err =                                  ms)) != 0)
    goto fa
    goto fa                                  HERE */
if ((err =                                  = 0)
    goto fa

err = sslRa
. . .
```



- Apple’s SSI
- **State mac e generated!**

GNU-TLS HELLO BUG

CVE-2014-3466 ...because SSL/TLS misery loves company!

```
- if (len < session_id_len) {
+ if (len < session_id_len || session_id_len >
TLS_MAX_SESSION_ID_SIZE) {
```

https://github.com/azet/CVE-2014-3466_PoC/blob/master/poc.py

http://radare.today/technical-analysis-of-the-gnutls-hello-vulnerability/

```
# PoC for CVE-2014-3466
# (gnutls: insufficient session id length check in _gnutls_read_server_hello)
#
# Author:   Aaron Zauner <azet@azet.org>
```

```
# Record Layer
R_Type          = '16'          # Handshake Protocol
R_Version       = '03 01'       # TLS 1.0
R_Length        = '00 fa'       # 250 Bytes

# Handshake Protocol: ServerHello
HS_Type         = '02'          # Handshake Type: ServerHello
HS_Length       = '00 00 f6'    # 246 Bytes
HS_Version      = '03 01'       # TLS 1.0
HS_Random       = '''
53 8b 7f 63 c1 0e 1d 72 0a b3 f8 a7 0f f5 5d 69
65 58 42 80 c1 fb 4f db 9a aa 04 a3 d3 4b 71 c7
'''                             # Random (gmt_unix_time + random bytes)
HS_SessID_Len   = 'c8'          # Session ID Length 200 Bytes (!)
HS_SessID_Data  = '''
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

```
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
'''                             # Session ID Data (Payload)
```

```
# Record Layer
R_Type          = '16'          # Handshake Protocol
R_Version       = '03 01'       # TLS 1.0
R_Length        = '00 fa'       # 250 Bytes

# Handshake Protocol: ServerHello
HS_Type         = '02'          # Handshake Type: ServerHello
HS_Length       = '00 00 f6'    # 246 Bytes
HS_Version      = '03 01'       # TLS 1.0
HS_Random       = '''
53 8b 7f 63 c1 0e 1d 72 0a b3 f8 a7 0f f5 5d 69
65 58 42 80 c1 fb 4f db 9a aa 04 a3 d3 4b 71 c7
'''                             # Random (gmt_unix_time + random bytes)
HS_SessID_Len   = 'c8'          # Session ID Length 200 Bytes (!)
HS_SessID_Data  = '''
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
'''                             # Session ID Data (Payload)

MaliciousServerHello = (
    R_Type      + R_Version     + R_Length          +
    HS_Type     + HS_Length     + HS_Version        +
    HS_Random   + HS_SessID_Len + HS_SessID_Data
).replace(' ', '').replace('\n', '').decode('hex')
```

```
# Record Layer
R_Type          = '16'          # Handshake
R_Version       = '03 01'       # TLS 1.0
R_Length        = '00 fa'       # 250 Bytes

# Handshake Protocol: ServerHello
HS_Type         = '02'          # Handshake
HS_Length       = '00 00 f6'    # 246 Bytes
HS_Version      = '03 01'       # TLS 1.0
HS_Random       = '''
53 8b 7f 63 c1 0e 1d 72 0a b3 f8 a7 0f f5 5d
65 58 42 80 c1 fb 4f db 9a aa 04 a3 d3 4b 71
'''                             # Random (gm
HS_SessID_Len   = 'c8'          # Session ID
HS_SessID_Data  = '''
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
'''                             # Session ID

MaliciousServerHello = (
    R_Type      + R_Version     + R_Length
    HS_Type     + HS_Length     + HS_Versio
    HS_Random   + HS_SessID_Len + HS_SessID_
).replace(' ', '').replace('\n', '').decode(
```

NESTED LENGTH FIELDS ARE DANGEROUS SYNTAX!

- Nested lengths are about data structure boundaries and nesting => they are **syntax**
- Length checks must be checked in the **parser**
  - e.g., if nested lengths do not agree the message is invalid
- Syntactically invalid messages should not be copied & processed
  - Semantic actions should wait until all syntax is checked
  - ...even if this means scanning message to the end

MORE MISERY! MS14-066

- MS SChannel: New code, same ASN.1 data.

```
001b:748598ce 0fb606          movzx   eax,byte ptr [esi]
001b:748598d1 0fb64e01        movzx   ecx,byte ptr [esi+1]  ds:0023:002cf29a=47
001b:748598d5 c1e008          shl     eax,8
001b:748598d8 03c1            add     eax,ecx
001b:748598da 8d4802          lea     ecx,[eax+2]
001b:748598dd 3b4d0c          cmp     ecx,dword ptr [ebp+0Ch]
001b:748598e0 77da            ja      schannel!CSsl3TlsServerContext::DigestCertVerify+0x196 (748598bc)
001b:748598e2 50              push    eax
001b:748598e3 83c602          add     esi,2
001b:748598e6 56              push    esi
001b:748598e7 ff75f4          push    dword ptr [ebp-0Ch]
001b:748598ea ff75f0          push    dword ptr [ebp-10h]
001b:748598ed ff75f8          push    dword ptr [ebp-8]
001b:748598f0 57              push    edi
001b:748598f1 e8b381ffff      call    schannel!CheckClientVerifyMessage (74851aa9)
001b:748598f6 eb3c            jmp     schannel!CSsl3TlsServerContext::DigestCertVerify+0x20e (74859934)
```

Command - Kernel 'com:port=com1,baud=115200' - WinDbg:6.2.9200.16384 X86

```
schannel!CSsl3TlsServerContext::DigestCertVerify+0x1a8:
001b:748598ce 0fb606          movzx   eax,byte ptr [esi]
1: kd> t
schannel!CSsl3TlsServerContext::DigestCertVerify+0x1ab:
001b:748598d1 0fb64e01        movzx   ecx,byte ptr [esi+1]
1: kd> d [esi]
002cf299  00 47 30 45 02 20 71 08-43 e4 0d 7d 92 9d 0b 15  .G0E. q.C..}....
002cf2a9  0e d5 77 59 64 b4 8d d9-29 00 4a be b0 ef e1 47  ..wYd...).J....G
002cf2b9  31 e2 28 24 43 96 02 21-00 a7 7d b2 05 75 69 94  1.($C..!..}..ui.
002cf2c9  93 58 ed b3 68 9f cb 3b-4[illegible] [illegible] c5 fa 02 6b 0f  .X..h..;Fm....k.
002cf2d9  a6 31 a3 11 eb c4 14 31-8c 14 03 01 00 01 01 16  .1.....1........
002cf2e9  03 01 00 30 42 38 54 2f-4e d6 ce cf 86 3d 18 46  ...0B8T/N....=.F
002cf2f9  16 69 be 70 6d 56 7d cd-7a 42 c9 d3 78 9d 47 f3  .i.pmV}.zB..x.G.
002cf309  fd e7 7b 8c ee c6 e2 f5-31 80 75 3d 29 bb a2 91  ..{.....1.u=)...
```

Total size of sig

Size of memcpy1

Size of memcpy2

http://www.securitysift.com/exploiting-ms14-066-cve-2014-6321-aka-winshock/

BERSERK!

- A variant of Bleichenbacher attack on PKCS#1 v.1.5 (CVE-2006-4339)

http://www.intelsecurity.com/advanced-threat-research/berserk.html

PARSER DIFFERENTIALS

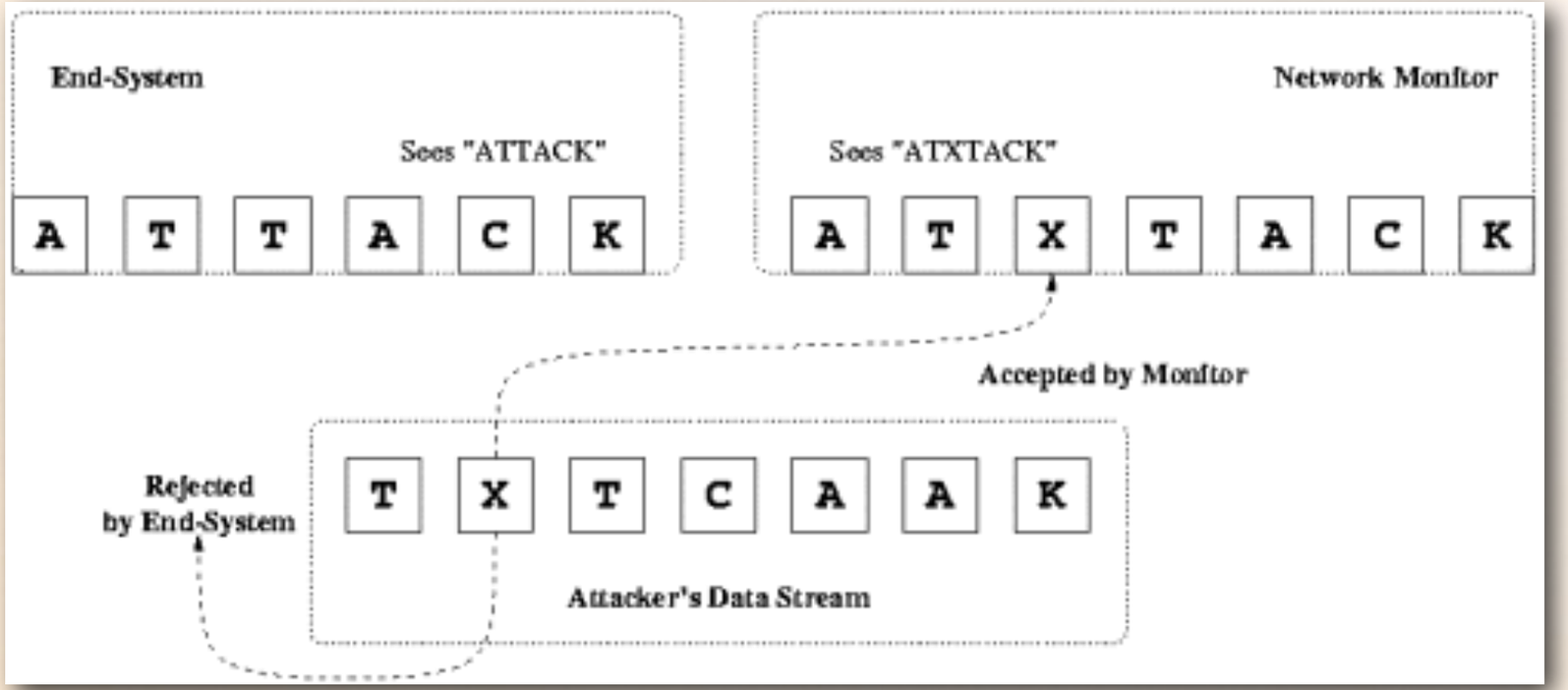
- Two parsers, one message ...

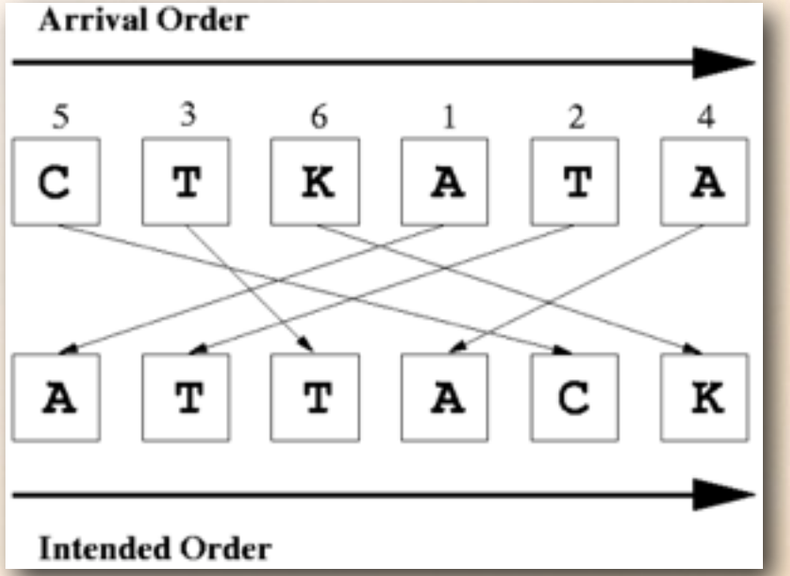
  two **different** parses!

- We've seen this before in:
  - "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Ptacek & Newsham, **1998**
  - X.509 certs: "PKI layer cake", Kaminsky, Sassaman, Patterson, **2010**

NIDS EVASION = PARSER DIFFS

- "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Ptacek, Newsham, 1998
  - Also Vern Paxson et al, 1999, protocol normalization



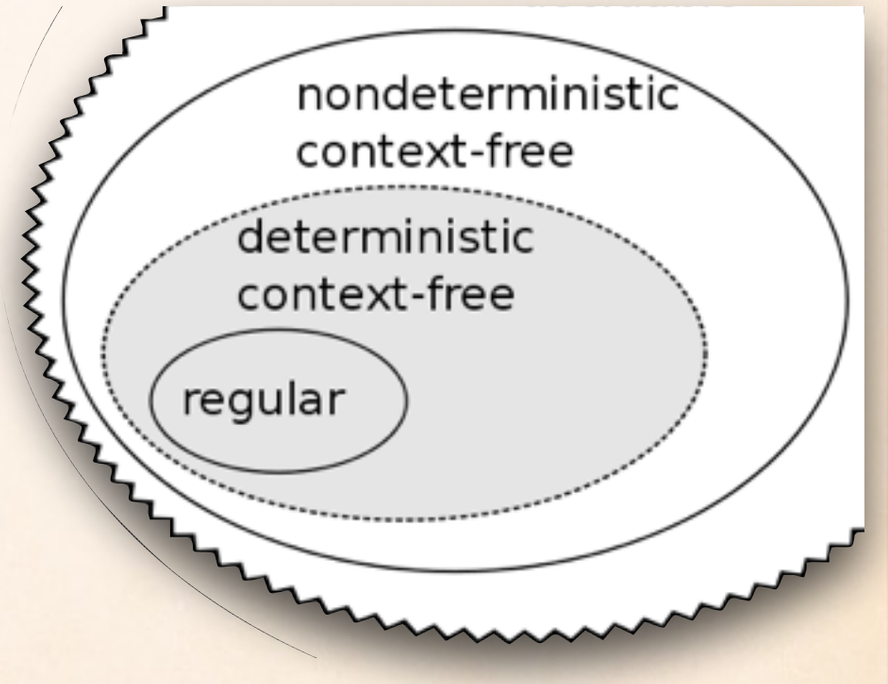


UNDECIDABLE PARSER DIFFERENTIALS

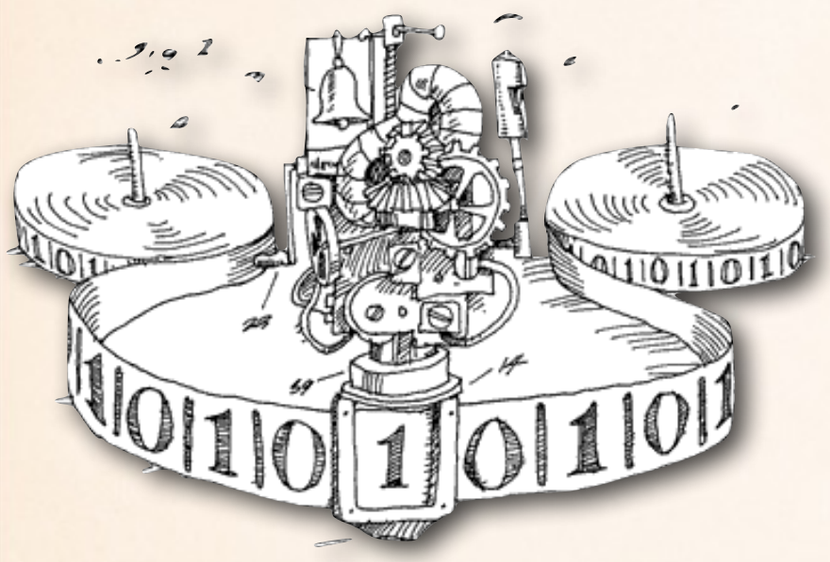
- "**PKI Layer Cake:** New Collision Attacks Against the Global **X.509** Infrastructure", Dan Kaminsky, Len Sassaman, Meredith L. Patterson, 2010

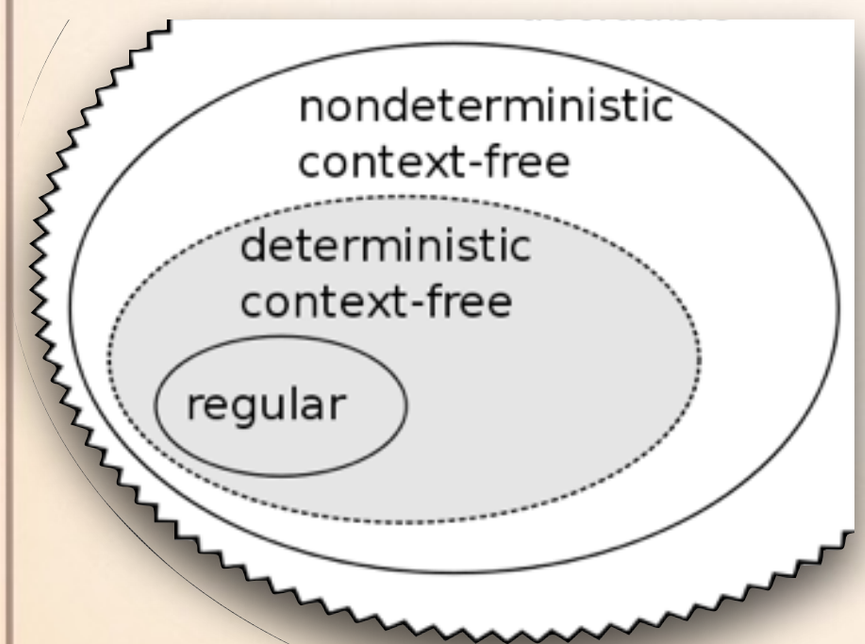
- X.509 / ASN.1 parsers disagree on what's in a common name (**CN**) => CA thinks it signs X, browser sees Y

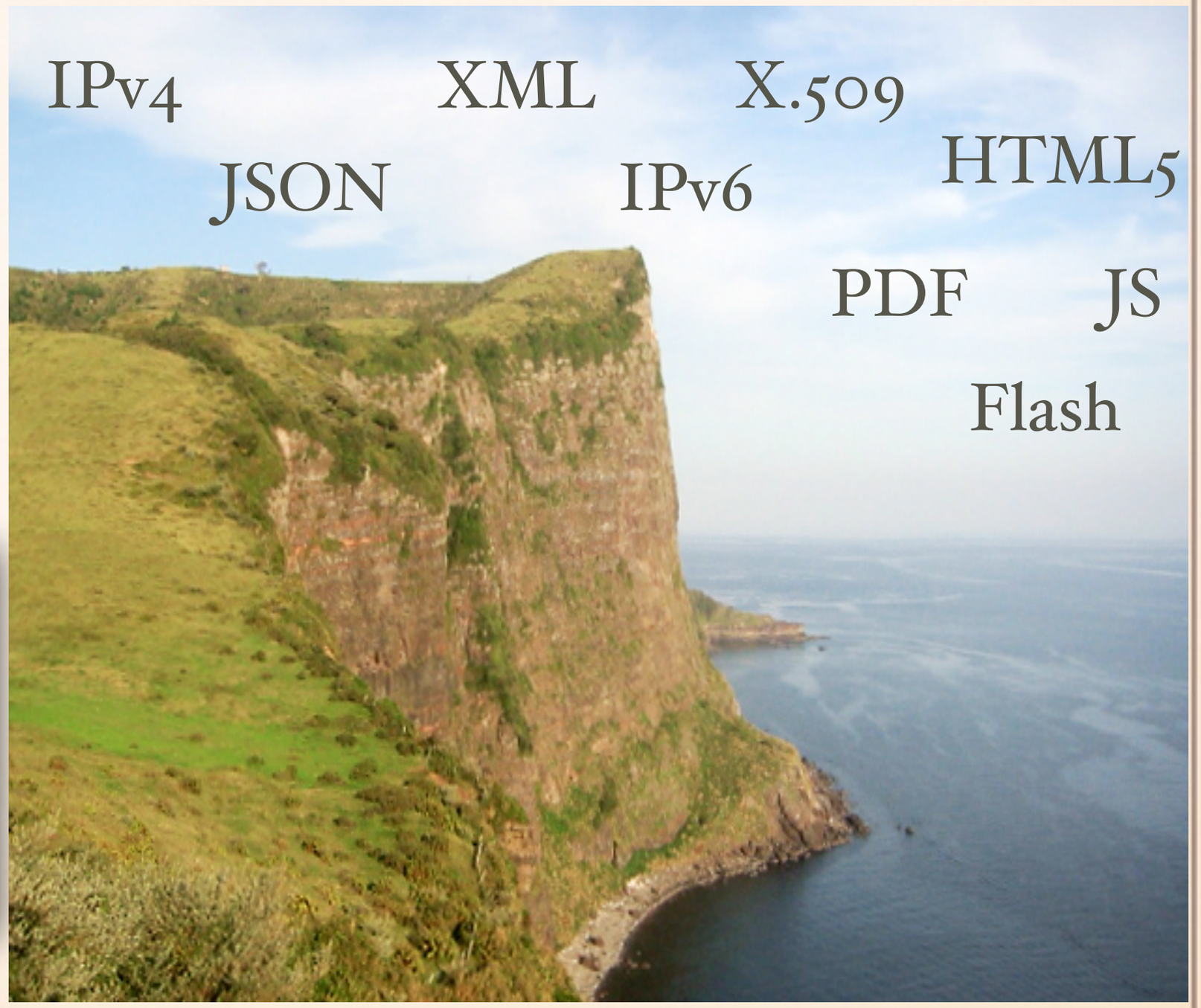
- Checking equivalence of parsers beyond deterministic context-free languages is **undecidable**



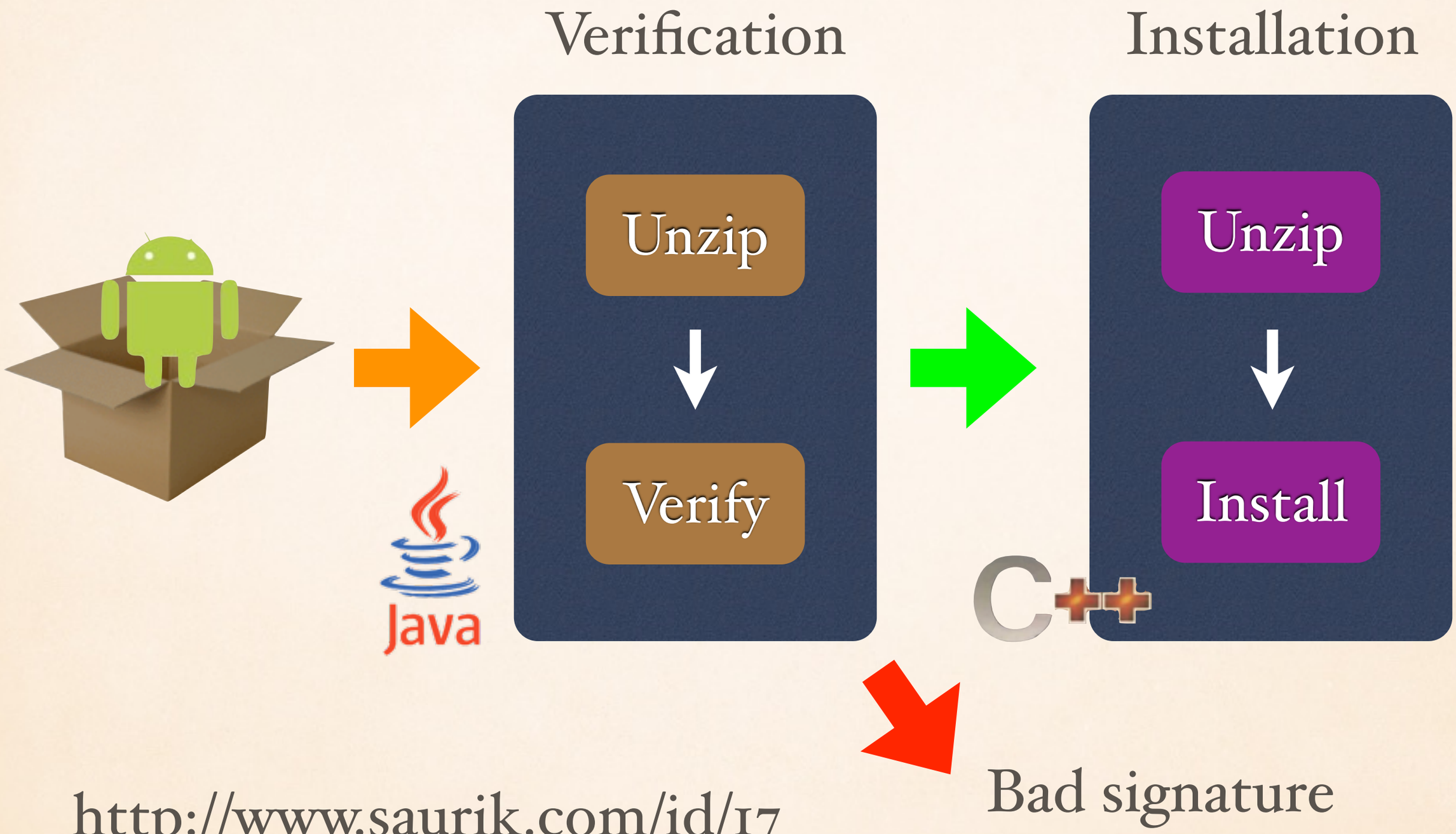
THE "UNDECIDABILITY CLIFF"







ANDROID MASTER KEY: A PARSER DIFFERENTIAL



http://www.saurik.com/id/17

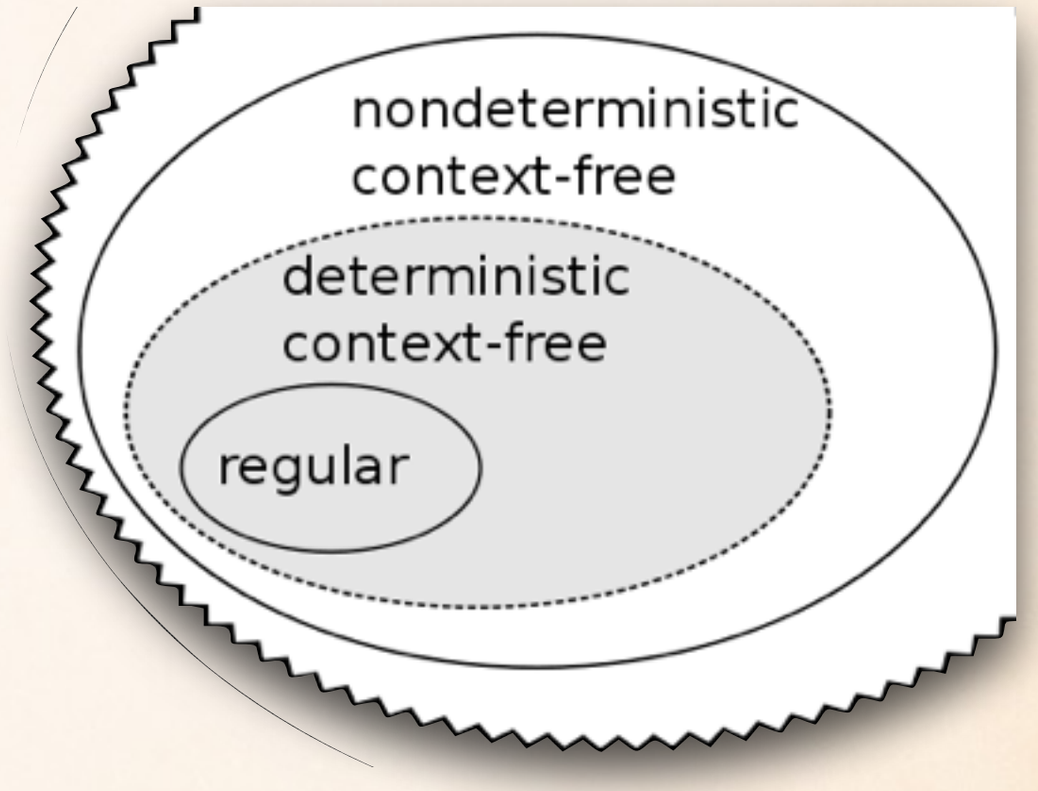
Android Master Key: A Parser Differential

- Android packages are signed & only installed if signature checks out
- **Java** crypto verifier followed by **C++** installer
- C++ has unsigned integers, Java doesn't => different results of unzipping
- Different contents "verified" vs installed

http://www.saurik.com/id/{17,18,19}

ANDROID MASTER KEY: A PARSER DIFFERENTIAL

- Initial fixes still kept two different parsers
  - Recipe for disaster: ***undecidable*** beyond deterministic context free languages



- Finally fixed right: the **same** parser used for both verification & installation, not two different parsers

ANDR SER

lation

zip

tall

http://ww re?

HTTP CHUNKED ENCODING

- Eliminates the need for *Content-Length* header
  - meant for cases where the size of HTTP response isn't known when response is started
  - e.g., unknown number of records fetched from a database

```
Transfer-Encoding: chunked
```

```
19
A bunch of data broken up
D
into chunks.
0
```

APACHE CVE-2002-3092

```
foreach my $offset (@offsets) {
    my $request;
    $request  = "GET / HTTP/1.1\r\n";
    $request .= "Host: $target_host:$target_port\r\n";
    $request .= "Transfer-Encoding: CHUNKED\r\n";
    $request .= "\r\n";
    $request .= "DEADBEEF ";

    # large nop sled plus shellcode
    $request .= $shellcode . "\r\n";

    # these three bytes are for address alignment
    $request .= "PAD";

    # place the appropriate amount of padding
    $request .= ("O" x $offset->[0]);

    # this is where ebx or esi points, make it jump over the return address
    $request .= "XX" . "\xeb\x04\xeb\x04";

    # this is the return address
    $request .= pack("V", $offset->[1]);
```

```
~~19~~ DEADBEEF
A bunch of data broken up
D
into chunks.
0
```

APACHE CVE-2002-3092

```
foreach my $offset (@offsets) {
    my $request;
    $request  = "GET / HTTP/1.1\r\n";
    $request .= "Host: $target_host:$target_port\r\n";
    $request .= "Transfer-Encoding: CHUNKED\r\n";
    $request .= "\r\n";
    $request .= "DEADBEEF ";

    # large nop sled plus shellcode
    $request .= $shellcode . "\r\n";
```

~~19~~ **DEADBEEF**

```
A bunch of data broken up
D
into chunks.
0
```

APACHE CVE-2002-3092

```
foreach my $offset (@offsets) {
    my $request;
    $request  = "GET / HTTP/1.1\r\n";
    $request .= "Host: $target_host:$target_port\r\n";
    $request .= "Transfer-Encoding: CHUNKED\r\n";
    $request .= "\r\n";
    $request .= "DEADBEEF ";
```

~~19~~ **DEADBEEF**

```
--- http_protocol.c.vuln        Fri Jun 14 16:12:50 2002
+++ http_protocol.c     Fri Jun 14 16:13:47 2002
@@ -2171,7 +2171,7 @@

     /* Otherwise, we are in the midst of reading a chunk of data */

-    len_to_read = (r->remaining > bufsiz) ? bufsiz : r->remaining;
+    len_to_read = (r->remaining > (unsigned int)bufsiz) ? bufsiz : r->
remaining;

     len_read = ap_bread(r->connection->client, buffer, len_to_read);
     if (len_read <= 0) {
```

Watch where you step!

AP… 092

```
foreach my
    my $re
    $reque
    $reque
    $reque
    $reque
    $reque
```

```
--- http_pro
+++ http_pro
@@ -2171,7 +

    /* Othe

-   len_to_
+   len_to_
remaining;

    len_rea
    if (len_
```

…\r\n";
…";

BEEF

…n up

… data */

…aining;
…ufsiz : r->

…to_read);

FAST FORWARD 11 YEARS...

- **Nginx** is found to have an exact **same** issue!

```
--- src/http/ngx_http_parse.c
+++ src/http/ngx_http_parse.c
@@ -2209,6 +2209,10 @@ data:

     }

+    if (ctx->size < 0 || ctx->length < 0) {
+        goto invalid;
+    }
+
     return rc;

 done:
```

```
    case sw_chunk_start:
        if (ch >= '0' && ch <= '9') {
            state = sw_chunk_size;
            ctx->size = ch - '0';
            break;
        }

        c = (u_char) (ch | 0x20);

        if (c >= 'a' && c <= 'f') {
            state = sw_chunk_size;
            ctx->size = c - 'a' + 10;
            break;
        }

        goto invalid;

    case sw_chunk_size:
        if (ch >= '0' && ch <= '9') {
            ctx->size = ctx->size * 16 + (ch - '0');
            break;
        }

        c = (u_char) (ch | 0x20);

        if (c >= 'a' && c <= 'f') {
            ctx->size = ctx->size * 16 + (c - 'a' + 10);
            break;
        }
```

```
2302 data:
2303
2304     ctx->state = state;
2305     b->pos = pos;
2306
2307     switch (state) {
2308
2309     case sw_chunk_start:
2310         ctx->length = 3 /* "0" LF LF */;
2311         break;
2312     case sw_chunk_size:
2313         ctx->length = 1 /* LF */
2314                       + (ctx->size ? ctx->size + 4 /* LF "0" LF LF */
2315                                    : 1 /* LF */);
2316         break;
2317     case sw_chunk_extension:
2318     case sw_chunk_extension_almost_done:
2319         ctx->length = 1 /* LF */ + ctx->size + 4 /* LF "0" LF LF */;
2320         break;
2321     case sw_chunk_data:
2322         ctx->length = ctx->size + 4 /* LF "0" LF LF */;
2323         break;
2324     case sw_after_data:
2325     case sw_after_data_almost_done:
2326         ctx->length = 4 /* LF "0" LF LF */;
2327         break;
2328     case sw_last_chunk_extension:
2329     case sw_last_chunk_extension_almost_done:
2330         ctx->length = 2 /* LF LF */;
2331         break;
2332     case sw_trailer:
2333     case sw_trailer_almost_done:
2334         ctx->length = 1 /* LF */;
2335         break;
2336     case sw_trailer_header:
2337     case sw_trailer_header_almost_done:
2338         ctx->length = 2 /* LF LF */;
2339         break;
2340
2341     }
2342
2343     if (ctx->size < 0 || ctx->length < 0) {
2344         goto invalid;
2345     }
```

STATE MACHINE DONE WRONG (AGAIN)

- **ngx_http_parse.c:**
  - 57 switch statements
  - 272 single-char case clauses
  - 2300+ SLOC
- States and inputs for all grammar elements all mixed together, **unintelligible**
- **Parser combinator style** would have exposed the issue immediately, not **10+ years** after the same bug in Apache

STA ONE

- **ngx_http_**
  - 57 switch
  - 272 single
  - 2300+ SL
- States and i mixed together, **u**
- Parser com e issue immediatel in Apache

FOR DESERT: SHELLSHOCK!

- **system("your command here")** actually means

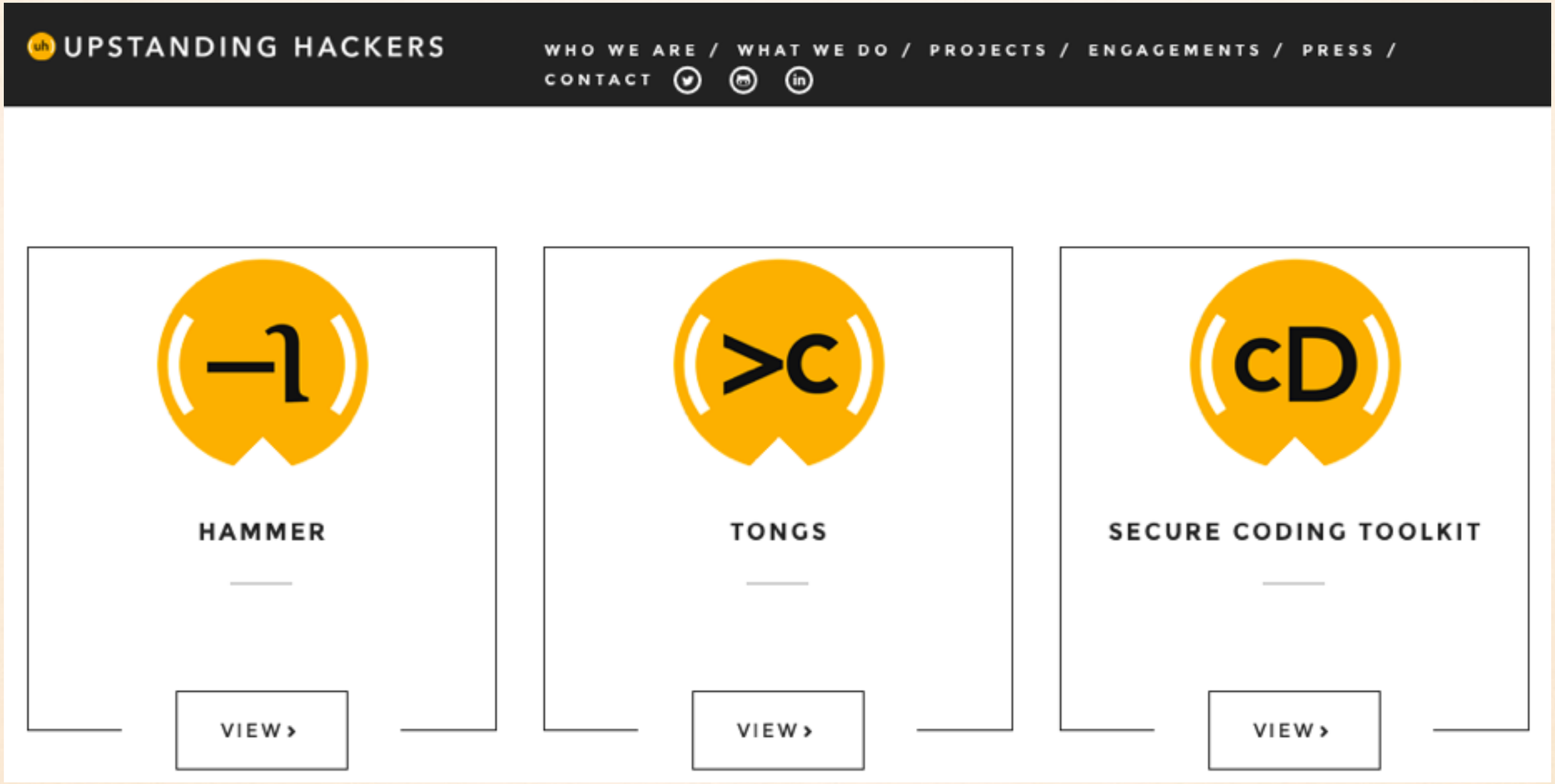
  parse_and_execute(ENV strings)

"Bash really is a local app that woke up one morning on the HMS CGI-BIN with a pounding headache"

- **Computation power exposed to external inputs is computation power given to attacker**

FOR DESERT: SHELLSHOCK!

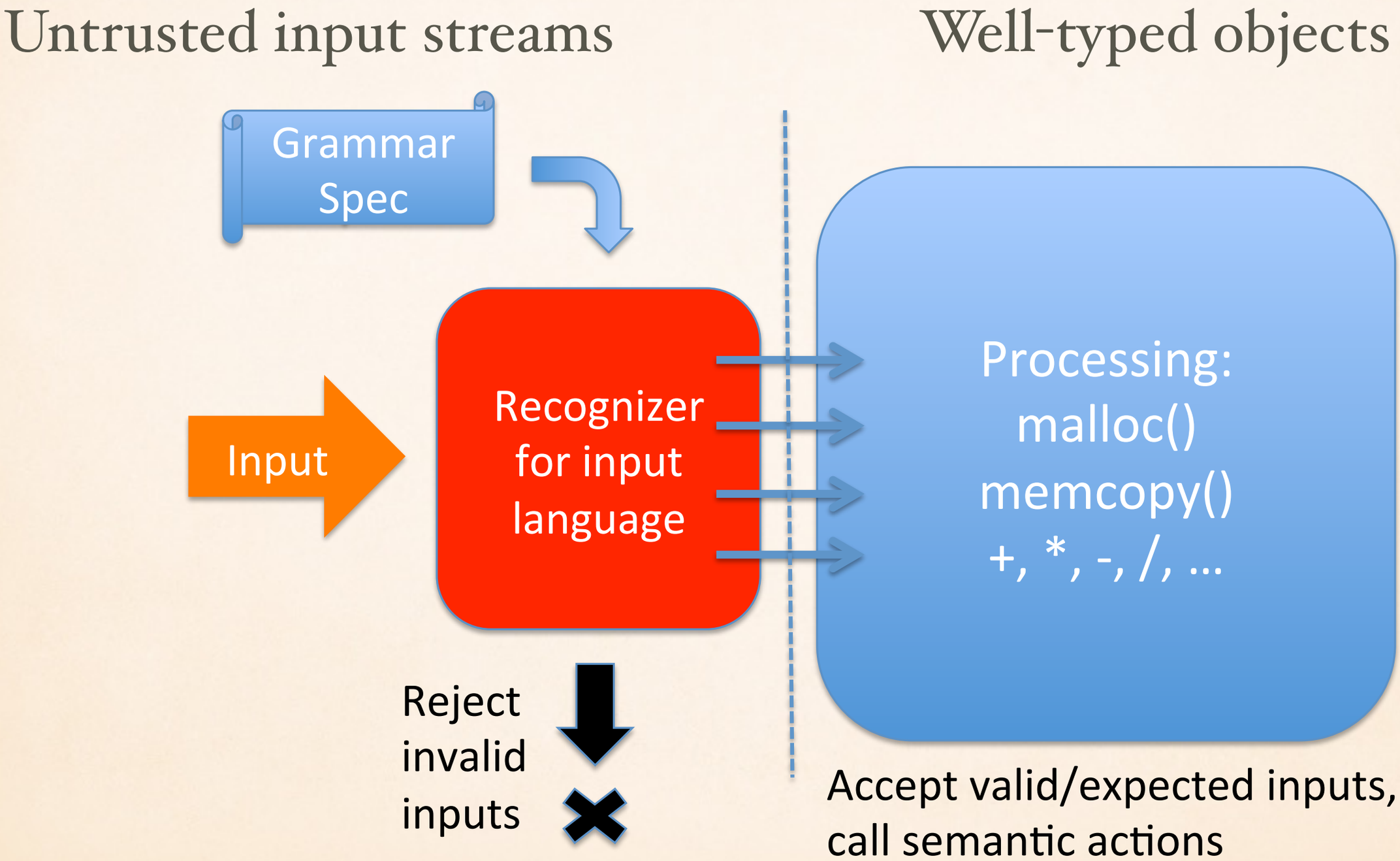
WHAT FUTURE HOLDS



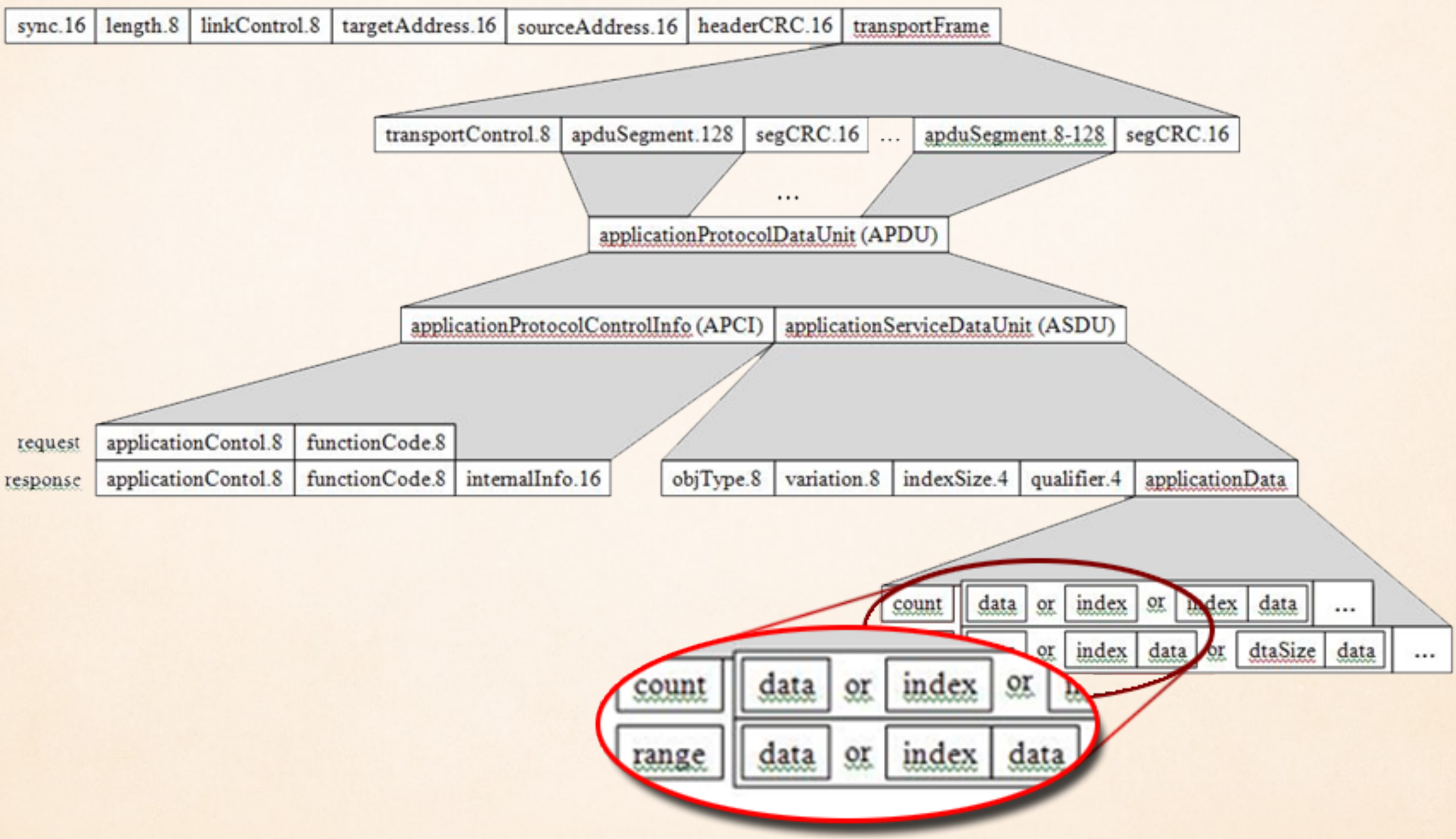
PARSER CONSTRUCTION

- Valid or expected inputs are a **language** & must be so treated
- Patch to Postel's principle: "[For security of your users], be **definite** about what you accept!"
- If you **hand-program** your parser, the **grammar** it expects/accepts must be **clear from the code**.
  - **Hammer**, a parser-combinator style kit for C/C++, Java, Python, .Net, Ruby, ...
    https://github.com/UpstandingHackers/hammer
    (Meredith L. Patterson et al)

PARSER CONSTRUCTION



PARSER-COMBINATOR STYLE: PARSERS ALL THE WAY DOWN



MAKE THE GRAMMAR THAT PARSER ACCEPTS CLEAR FROM THE CODE!

```
05 64 14 F3
01 00 00 04
0A 3B C0 C3
01 3C 02 06
3C 03 06 3C
04 06 3C 01
06 9A 12
```

```
start = h_token("\x05\x64");
len = h_int_range(h_uint8(), 5, 255);
ctrl = h_uint8();
dst = h_uint16();
src = h_int_range(h_uint16(), 0, 65519);
crc = h_uint16();
hdr = h_attr_bool(h_sequence(h_ignore(start),
        len, ctrl, dst, src, crc, NULL),
        validate_crc);
frame = h_attr_bool(h_sequence(hdr,
        h_optional(transport_frame),
        h_end_p(), NULL),validate_len);
```

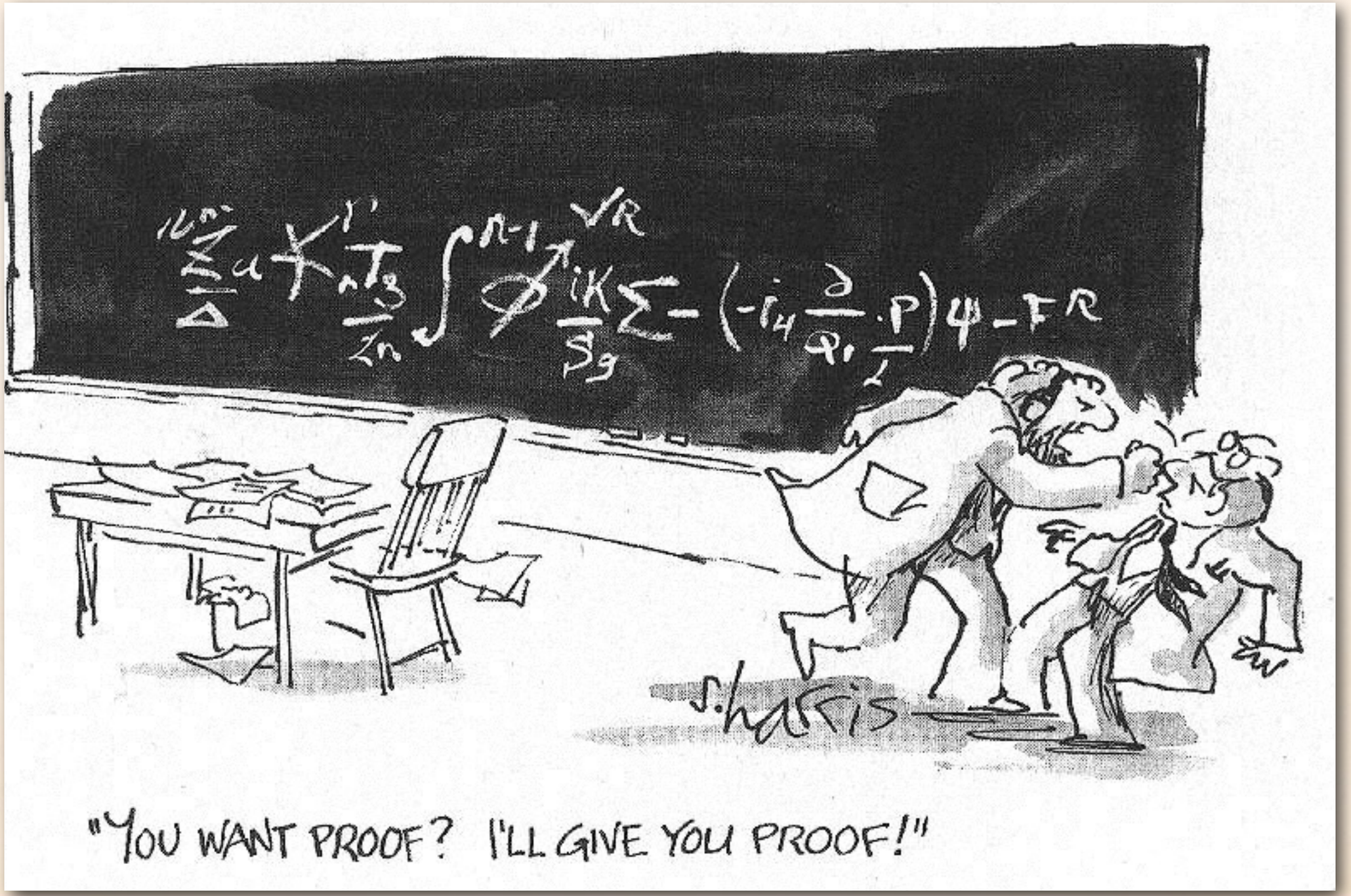
AUDITING WITH LANGSEC

- Practical rules for input-language decisions: which to choose?
  - **JSON** vs. XML vs. ASN.1
  - **DER** vs. BER
- Auditing of input-handling code
  - *"Where is your **recognizer**?"*
  - *"Do you really need **recursive nesting** syntax/ **cross-layer** context dependency/ **cross-object** dependency?*

CVEs:

XML	JSON
635 (170 XXE)	58

PROOFS TO THE RESCUE?



An Axiomatic Basis for Computer Programming

C. A. R. Hoare
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

KEY WORDS AND PHRASES: axiomatic method, theory of programming' proofs of programs, formal language definition, programming language design, machine-independent programming, program documentation
CR CATEGORY: 4.0, 4.21, 4.22, 5.20, 5.21, 5.23, 5.24

Volume 12 / Number 10 / October, 1969
Communications of the ACM

AB OVO

- Proving correctness of programs ***deductively***, from axioms
- "..axioms offer a simple and flexible technique for leaving certain aspects of a language ***undefined*** ... [which is] absolutely essential for standardization purposes."

C.A.R. HOARE, 1968..

Thus the practice of proving programs would seem to lead to solution of three of the most pressing problems in software and programming, namely, reliability, documentation, and compatibility. However, program proving, certainly at present, will be difficult even for programmers of high caliber; and may be applicable only to quite simple program designs. As in other areas, reliability can be purchased only at the price of simplicity.

An Axiomatic Basis for Computer Programming

C. A. R. Hoare 1969

$$\mathbb{P}\ \{\ \mathbb{Q}\ \}\ \mathbb{R}$$

$$P\ \{Q\}\ R$$

Precondition Code Result

D1 Rules of Consequence

If $P\{Q\}R$ and $R \supset S$ then $P\{Q\}S$

If $P\{Q\}R$ and $S \supset P$ then $S\{Q\}R$

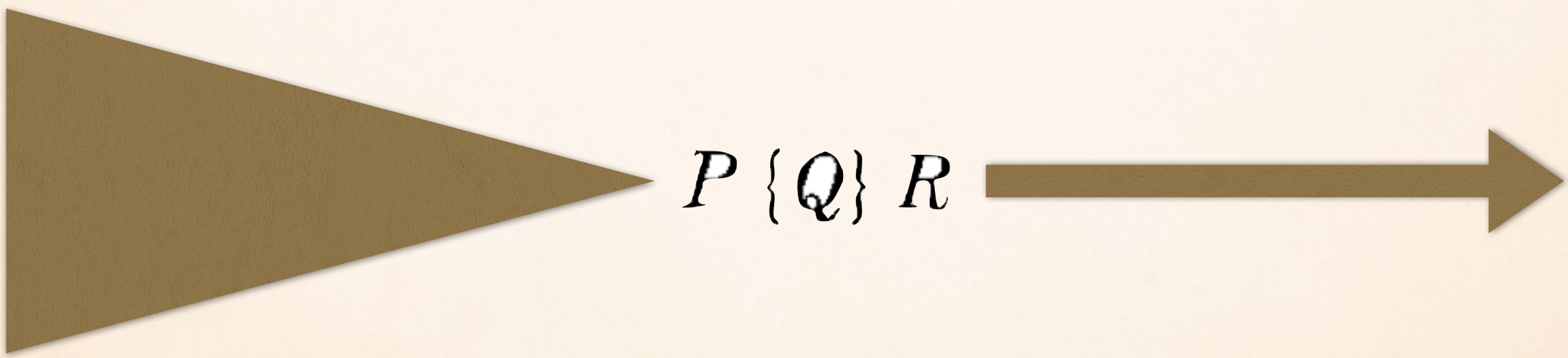
D2 Rule of Composition

If $P\{Q_1\}R_1$ and $R_1\{Q_2\}R$ then $P\{(Q_1\ ;\ Q_2)\}R$

ENTER WEIRD MACHINES

Assume Q is proven correct, P { Q } R

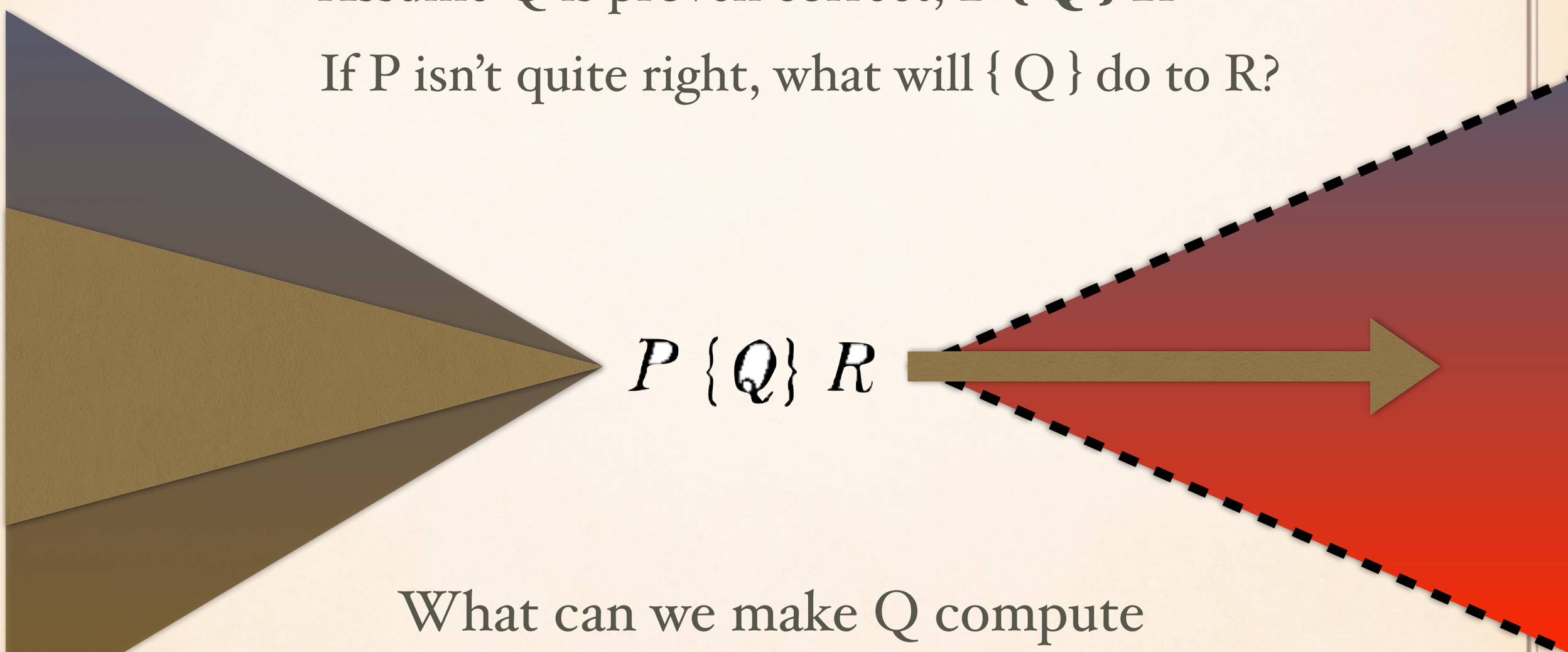
If P isn't quite right, what will { Q } do to R?



ENTER WEIRD MACHINES

Assume Q is proven correct, **P { Q } R**

If P isn't quite right, what will { Q } do to R?



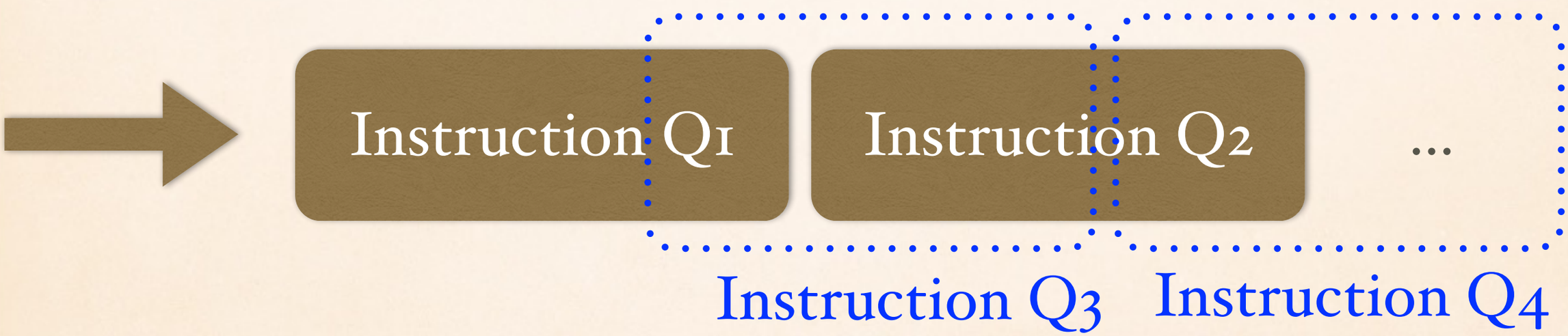
What can we make Q compute
by varying inputs it **wasn't** verified for?

ABSTRACTION VS COMPOSITION

D2 Rule of Composition

If $P\{Q_1\}R_1$ and $R_1\{Q_2\}R$ then $P\{(Q_1 ; Q_2)\}R$

- So you put together { Q1 ; Q2 }. How many programs did you actually create?



Coq: The world's best macro assembler?

Andrew Kennedy Nick Benton
Microsoft Research

Jonas B. Jensen
ITU Copenhagen

Pierre-Evariste Dagand
University of Strathclyde

```
Definition call_cdecl3 f arg1 arg2 arg3 :=
  PUSH arg3;; PUSH arg2;; PUSH arg1;;
  CALL f;; ADD ESP, 12.

Definition main (printfSlot: DWORD) :=
  (* Argument in EBX *)
  letproc fact :=
    MOV EAX, 1;;
    MOV ECX, 1;;
      (* while ECX <= EBX *)
      while (CMP ECX, EBX) CC_LE true (
        MUL ECX;; (* Multiply EAX by ECX *)
        INC ECX
      )
  in
    LOCAL format;
      MOV EBX, 10;; callproc fact;;
      MOV EDI, printfSlot;;
      call_cdecl3 [EDI] format EBX EAX;;
      MOV EBX, 12;; callproc fact;;
      MOV EDI, printfSlot;;
      call_cdecl3 [EDI] format EBX EAX;;
      RET 0;;
    format:;;
      ds "Factorial of %d is %d";; db #10;; db #0.

Compute bytesToHex
  (assemble #x"C0000004" (main #x"C0000000")).
```

- Coq, the proof assistant than can do induction proofs in $\mathbb{N}$
- Bit-level models of x86 instructions + mnemonics
- Verified assembly language
  - Also, see **Ironclad**, Hawblitzel et al., OSDI'14

EXPLOITATION IS VERIFICATION

BY THANASSIS AVGERINOS, SANG KIL CHA, ALEXANDRE REBERT, EDWARD J. SCHWARTZ, MAVERICK WOO, AND DAVID BRUMLEY

Automatic Exploit Generation

AEG is far from being solved. Scalability will always be an open and interesting problem. As of February 2013, AEG tools typically scale to finding buffer overflow exploits in programs the size of common Linux utilities.

Our research team and others cast AEG as a program-verification task but with a twist (see the sidebar "History of AEG"). Traditional verification takes a program and a specification of safety as inputs and verifies the program satisfies the safety specification. The twist is we replace typical safety properties with an "exploitability" property, and the "verification" process becomes one of finding a program path where the exploitability property holds. Casting AEG in a verification framework ensures AEG techniques are based on a firm theoretic foundation. The verification-based approach guarantees sound analysis, and automatically generating an exploit provides proof that the reported bug is security-critical.

"HOUSTON, WE HAVE A PROBLEM"

- Wassenaar Arrangement (Dec. 2013) defines "***intrusion software***"
  - "...The **modification** of the **standard execution path** of a program or process in order to allow the execution of externally provided instructions..."
  - Controls means of **generating**, **developing**, **operating** "intrusion software"
- Inputs become **regulated arms?**
  - More in our "**Information Security War Room**" invited talk with FX at USENIX Security 2014

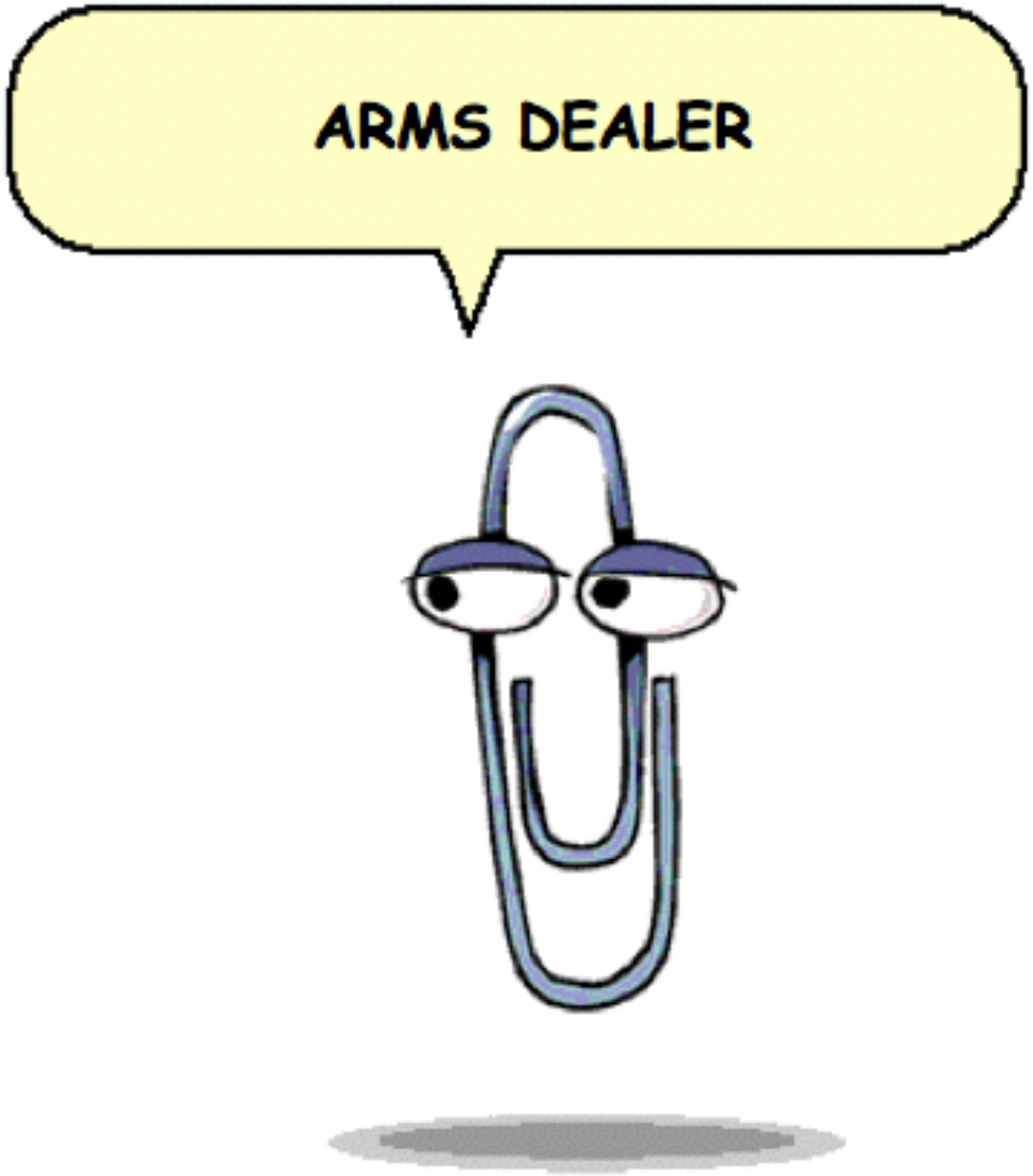


Image credit: FX

RECOMMENDATIONS

- Specify your valid & expected input with **a grammar**
  - Keep the **input language** as **simple** as possible
- If you hand-write the parser, make sure **the grammar is obvious from code**
  - Use parser combinator style! (e.g., Hammer)
- Don't mix **semantic actions** with **syntax recognition**!
  - ***"Full recognition before processing"***
  - Careful with ***memcopy***, etc. before input is fully validated!

RECOMMENDATIONS

- Trustworthiness must at **least** include **constraining** & **isolating** emergent computation ("weird machines")
- **Co-design data** formats **&** their parsing **code** to have least complexity, to make verification tractable
  - The only way to avoid complexity cliff

LANGSEC VIEW OF CWE

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

CWSS CWRAF

Presentation Filter: Mapping-Friendly

Improper Input Validation

Weakness ID: 20 *(Weakness Class)* 2009 CWE/SANS Top 25 **Status:** Usable

Description

Description Summary

The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program.

Extended Description

When software does not validate input properly, an attacker is able to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution.

ParentOf	B	15	External Control of System or Configuration Setting	700
ParentOf	C	21	Pathname Traversal and Equivalence Errors	699
ParentOf	C	73	External Control of File Name or Path	699
ParentOf	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	
ParentOf	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	700
ParentOf	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	700
ParentOf	B	99	Improper Control of Resource Identifiers ('Resource Injection')	700
ParentOf	C	100	Technology-Specific Input Validation Problems	699
ParentOf	V	102	Struts: Duplicate Validation Forms	700
ParentOf	V	103	Struts: Incomplete validate() Method Definition	700
ParentOf	V	104	Struts: Form Bean Does Not Extend Validation Class	700
ParentOf	V	105	Struts: Form Field Without Validator	700 1000
ParentOf	V	106	Struts: Plug-in Framework not in Use	700 1000

2010/2011 CWE/SANS Top 25

ParentOf	V	107	Struts: Unused Validation Form	700
ParentOf	V	108	Struts: Unvalidated Action Form	700 1000
ParentOf	V	109	Struts: Validator Turned Off	700 1000
ParentOf	V	110	Struts: Validator Without Form Field	700
ParentOf	B	111	Direct Use of Unsafe JNI	699 700
ParentOf	B	112	Missing XML Validation	699 700 1000
ParentOf	B	113	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')	700
ParentOf	B	114	Process Control	699 700 1000
ParentOf	B	117	Improper Output Neutralization for Logs	700
ParentOf	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	699 700
ParentOf	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	700
ParentOf	B	129	Improper Validation of Array Index	699 1000
ParentOf	B	134	Uncontrolled Format String	700
ParentOf	B	170	Improper Null Termination	700
	B	190	Integer Overflow or Wraparound	700
	B	466	Return of Pointer Value Outside of Expected Range	700
ParentOf	B	470	Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')	699 700
ParentOf	V	554	ASP.NET Misconfiguration: Not Using Input Validation Framework	699 1000
ParentOf	V	601	URL Redirection to Untrusted Site ('Open Redirect')	699
ParentOf	B	606	Unchecked Input for Loop Condition	699 1000
ParentOf	V	622	Improper Validation of Function Hook Arguments	699 1000
ParentOf	V	626	Null Byte Interaction Error (Poison Null Byte)	699 1000
ParentOf	CO	680	Integer Overflow to Buffer Overflow	1000
ParentOf	CO	690	Unchecked Return Value to NULL Pointer Dereference	1000
ParentOf	CO	692	Incomplete Blacklist to Cross-Site Scripting	1000
ParentOf	V	781	Improper Address Validation in IOCTL with METHOD_NEITHER I/O Control Code	699 1000
ParentOf	V	785	Use of Path Manipulation Function without Maximum-sized Buffer	699 700
ParentOf	V	789	Uncontrolled Memory Allocation	1000

Large classes of weaknesses…

Brief Listing of the Top 25

The Top 25 is organized into three high-level categories that contain multiple CWE entries.

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output
- CWE-89: Failure to Preserve SQL Query Structure ('SQL Injection')
- CWE-79: Fail
- CWE-78: Imp ('OS Commar
- CWE-319: Cl
- CWE-352: Cr
- CWE-362: Ra
- CWE-209: Er

2009 CWE/SANS Top 25

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

For each weakness, its ranking in the general list is provided in square brackets.

Rank	CWE ID	Name
[1]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[2]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[4]	CWE-352	Cross-Site Request Forgery (CSRF)
[8]	CWE-434	Unrestricted Upload of Fil
[9]	CWE-78	Improper Neutralization c Command Injection')
[17]	CWE-209	Information Exposure Th
[23]	CWE-601	URL Redirection to Untrus
[25]	CWE-362	Race Condition

2010 CWE/SANS Top 25

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

For each weakness, its ranking in the general list is provided in square brackets.

Rank	CWE ID	Name
[1]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[4]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[9]	CWE-434	Unrestricted Upload of File with Dangerous Type
[12]	CWE-352	Cross-Site Request Forgery (CSRF)
[22]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')

2011 CWE/SANS Top 25
(and still current)

…are failures of recognition!

LANGSEC WORKSHOP 2015

- Second year of the **LangSec workshop** at the IEEE Security & Privacy Symposium
- **http://spw15.langsec.org/ -- Thu May 21, 2015**