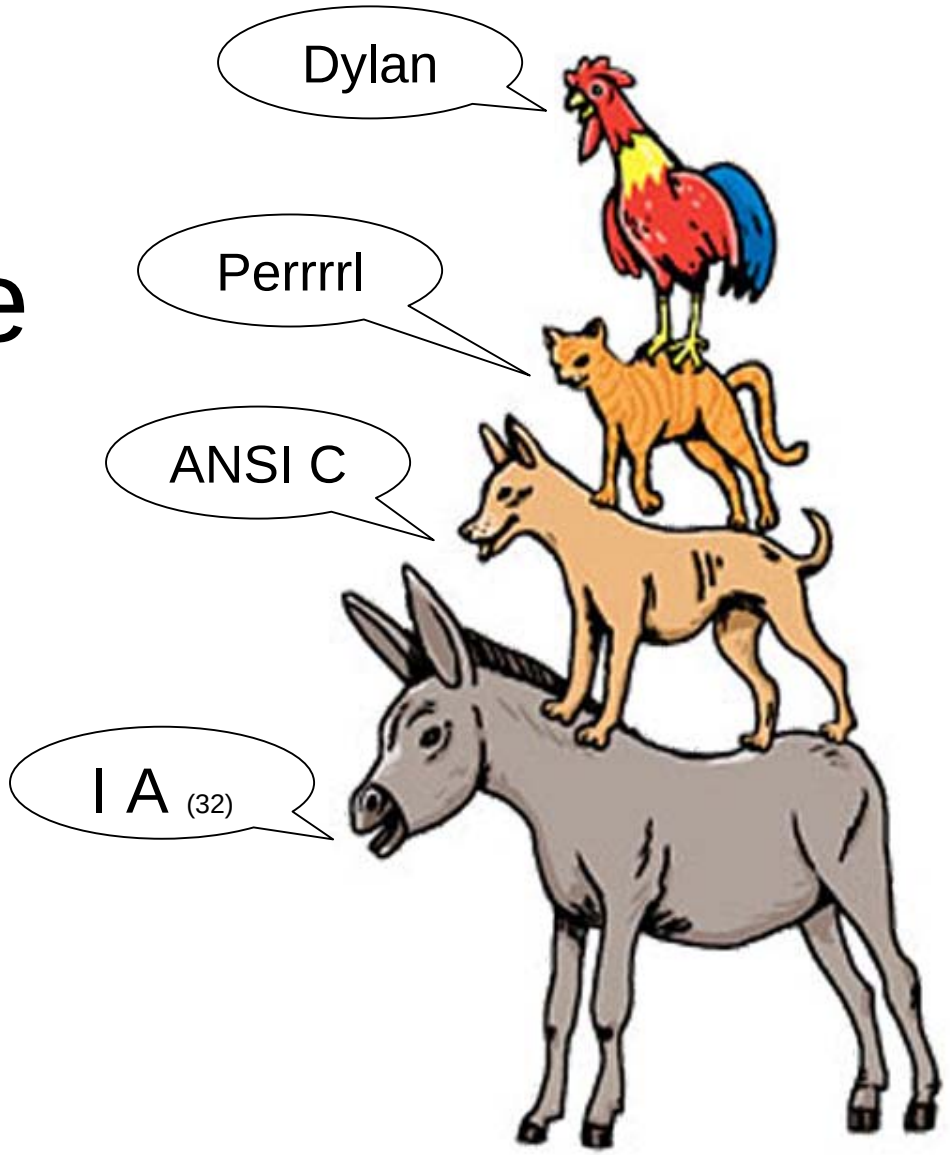
Das Literarische Code-Quartett



Andreas Bogk
Lisa Thalheim
Felix von Leitner
FX of Phenoelit

Wie versprochen: Postgres

postgresql/interfaces/ecpg/preproc/preproc.y:

```
char *dimension = $3.index1;
char dim[14L];
[...]
        sprintf(dim, "[%s]", dimension);
```

Premature abstraction is the root of all evil

```
unsigned long get(size_t start, size_t len) const {
        unsigned long dest = 0;
        std::bitset<N> tmp(m_bits);
        int i;
        for(unsigned i=0; i<start; ++i) {
            tmp.reset(i);
        }
        for(i=start+len; i<N; ++i) {
            tmp.reset(i);
        }
        tmp = tmp >> start;
        dest = tmp.to_ulong();
        return dest;
    }
```

C als High Level Assembler

```
if (foo)
  a=b;
else
  a=c;
```

➔

```
a=((-foo)&b) | (~-foo)&c;
```

for this case default to break

```
void print_joined (const char *first, const char *delim,
                   const char *last, const char *def,
                   char **list) {
   switch (*list == NULL) {
     for (; *list; ++list) {
       printf("%s", delim);
       if (0) {
   case 0:
         printf("%s", first);
       }
       printf("%s", *list);
     }
     printf("%s", last);
     break;
   default:
     printf("%s", def);
     break;
   }
 }
```

Kontrollflusssalat ist gesund

```
checkifmore:
    pCancel = NULL;
    if (!IsListEmpty(&Queue))
    {
        fSleep = TRUE;
    }
    pListElem = &Queue;
    while ((pListElem = pListElem->Flink) != &Queue)
    {
        pContext = GET_RECORD(pListElem, CONTEXT, Link);

        pOriginal = pContext->pOriginal;
        SetCancelRoutine(pOriginal, NULL);

        pCancel = pContext->pExtended;
        pContext->pExtended = NULL;
        goto cancelqueued;
    }

cancelqueued:
    if (pCancel != NULL)
    {
        BOOLEAN     fCancelled = FALSE;
        fCancelled = pCancel->Cancel;
        if (fCancelled == FALSE)
        {
            Cancel(pCancel);
        }

        goto checkifmore;
    }
```

Defines from hell

Nvidia-PCI-Express bug fix:

```
-#define PCIGETMEMORY64HIGH(b)  (*((CARD32*)&b + 1))
+#define PCIGETMEMORY64HIGH(b)  (*((CARD32*)&(b) + 1))
```

Defines from hell

Immer schön portierbar bleiben:

```
#define SwapWord( theWord )     (theWord) + 0
#define SwapDWord( theDWord )   (theDWord) + 0
```

Defines from hell

Sehr beliebt auch:

```
#define offsetof(type,member) \
   ((size_t) &((type*)0)->member)
```

Defines from hell

```
#define NEW_FAST_TEMP_STR(NAME, EstimatedBiggestSize, LenNeeded)\
    char    __##NAME##__StaticVersion[EstimatedBiggestSize]; \
    char*   NAME;                                            \
    UINT32  ulNeeded##NAME##Len = (LenNeeded);               \
                                                             \
    if (ulNeeded##NAME##Len <= EstimatedBiggestSize)\
    {                                                        \
        NAME = __##NAME##__StaticVersion;                    \
    }                                                        \
    else                                                     \
    {                                                        \
        NAME = new char[ulNeeded##NAME##Len];                \
    }

#define DELETE_FAST_TEMP_STR(NAME)                  \
    if (NAME != __##NAME##__StaticVersion)          \
    {                                               \
        delete[] NAME;                              \
    }
```

C++? Niemals!

"… nur weil man's nicht besser kennt, ist es noch lange nicht egal" – Kettcar

```
char *thc_create_ipv6(
    char *interface, int prefer, int *pkt_len,
    unsigned char *src, unsigned char *dst,
    int ttl, int length, int label,
    int class, int version) {
  ...
  if (class == 0)
    hdr->class = 0;
  else
    hdr->class = class;
```

Ein Javanese im C++-Land

```
char *error_string=new char[512];
```

Internationalisierungsbestrebungen

Internationalisierung Marke Postgres:

(oder: Die Über-Übersetzung)

```
LANG=de → "owner" → "eigent"umer"
   ➔ SQL parse error
```

I ♥ UNICODE !

... das machen wir mal in einer extra Funktion, wegen der Sicherheit und so ...

```
private DWORD MaxResultCharacters (byte tag, DWORD cchString_)
{
    SafeInt<DWORD> cchString = cchString_;
   //
   // Determine maximum number of characters needed to hold a string
   //
   // String is:           Want Unicode     Want Ansi
   // -----------------    ------------     ---------
   // unicode                   n               2n
   // compressedUnicode         n               2n
   // DBCS                      n               n
   //
   return (cchString + 1);
}
```

Zeigermagie

```
p--;
while (p >= image) {
   if (*p != '.') {
      if (*p != '9') {
         *p += 1;  return(0);
      }
      *p = '0';
   }
   p--;
}
```

Die Hoffnung stirbt zuletzt

```
/* this function is a huge hack.  since all of the fault types have the
same base fault (wsbf_BaseFaultType), they'll all start with the same
members. hopefully the compiler always aligns the first members the same.
 */

globus_object_t *
globus_i_gram_fault_to_error(
    wsgram_FaultResourcePropertyType *  gram_fault)
{
    globus_object_t *                   error = NULL;
    wsbf_BaseFaultType *                fault;

    if(gram_fault->choice_value.type !=
        wsgram_FaultResourcePropertyType_undefined)
    {
        fault = (wsbf_BaseFaultType *)
                  &gram_fault->choice_value.value.fault;
        error = globus_error_convert_wsrf_fault(fault);
    }

    return error;
}
```

Zeigermagie

```
do{*from++=*to++;}while{--count>0};
```

Zwischenablage bei RIM

Hmm, was haben wir denn noch frei? Ach ja, dieses timeout von dem select() Zeugs, das können wir ja nehmen.

```
lea     esi, [edi+6]                    ; ESI = length
cmp     esi, [ebx+Class.length]
mov     [ebp+timeout.tv_usec], esi
```

String 2 Int, die ewige Story

```
static unsigned int ReadNewsProfile(char *text,long int length,Image
    *image, int type)
{
  register unsigned char   *p;
  [...]

#if defined(GET_ONLY_IPTC_DATA)
  /*
    Eat OSType, IPTC ID code, and Pascal string length bytes.
  */
  p+=6;
  length=(*p++);
  if (length)
    p+=length;
  if ((length & 0x01) == 0)
    p++;  /* align to an even byte boundary */
  length=(p[0] << 24) | (p[1] << 16) | (p[2] << 8) | p[3];
  p+=4;
#endif
  return SetImageProfile(image,"8BIM",p,(size_t) length);
}
```

No, really?

```
// XXX untested!

int i;
char *error_string=NULL;

sprintf(error_string,
  "Foo::delEntry \nARP: IP in Table is:
   %i.%i.%i.%i",
  arp_pa[0],arp_pa[1],arp_pa[2],arp_pa[3]);
```

...

```
char *error_string;

uint8_t *newMAC;
uint8_t newMACvector[6];
uint8_t newIPvector[4];


error_string = new char[255];

...

free(error_string);
p->kill();
delete (error_string);
// output(0).push(q);
return(q);
```

Linux SATA

Linux in die Rechenzentren! Der Kernel ist von höchster Qualität.

```
static void pdc_host_init(unsigned int chip_id, struct ata_probe_ent *pe)
{
        void *mmio = pe->mmio_base;
        u32 tmp;

        /*
         * Except for the hotplug stuff, this is voodoo from the
         * Promise driver.  Label this entire section
         * "TODO: figure out why we do this"
         */

        /* change FIFO_SHD to 8 dwords, enable BMR_BURST */
        tmp = readl(mmio + PDC_FLASH_CTL);
        tmp |= 0x12000; /* bit 16 (fifo 8 dw) and 13 (bmr burst?) */
        writel(tmp, mmio + PDC_FLASH_CTL);

        /* clear plug/unplug flags for all ports */
        tmp = readl(mmio + PDC_SATA_PLUG_CSR);
        writel(tmp | 0xff, mmio + PDC_SATA_PLUG_CSR);
```

Linux SATA die 2.

Wie geht denn ein Flush? Hmm ... grbl ... murmel… Na warten wir einfach. Ist ja open source, kann ja ein anderer fixen.

```
/* software reset.  causes dev0 to be selected */
if (ap->flags & ATA_FLAG_MMIO) {
  writeb(ap->ctl, (void __iomem *) ioaddr->ctl_addr);
  udelay(20);     /* FIXME: flush */
  writeb(ap->ctl | ATA_SRST, (void __iomem *) ioaddr->ctl_addr);
  udelay(20);     /* FIXME: flush */
  writeb(ap->ctl, (void __iomem *) ioaddr->ctl_addr);
```

Scapy

```
class HSRP(Packet):
    name = "HSRP"
    fields_desc = [
        ByteField("version", 0),
        ByteEnumField("opcode", 0, { 0:"Hello",
                                     1:"Coup",
                                     2:"Resign",
                                     3:"Advertise"}),
        ByteEnumField("state", 16, { 16:"Active"}),
        ByteField("hellotime", 3),
        ByteField("holdtime", 10),
        ByteField("priority", 120),
        ByteField("group", 1),
        ByteField("reserved", 0),
        StrFixedLenField("auth","cisco",8),
        IPField("virtualIP","192.168.1.1") ]
```

Ethereal

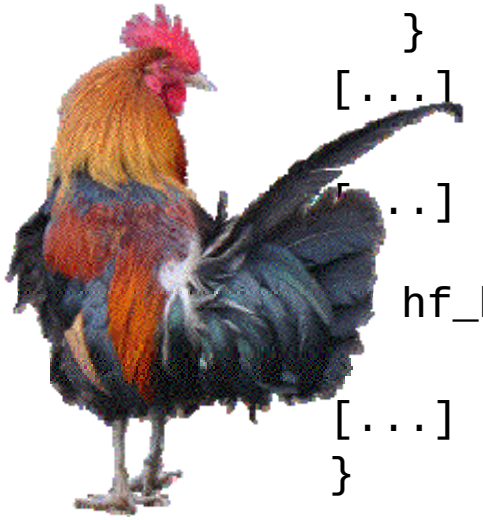
```
static gint hf_hsrp_opcode = -1;

struct hsrp_packet {
/*Multicast to 224.0.0.2, TTL 1, UDP, port 1985 */
[...]
        guint8  opcode;
[...]
};
```

Ethereal

```
#define HSRP_OPCODE_HELLO     0
#define HSRP_OPCODE_COUP      1
#define HSRP_OPCODE_RESIGN    2
#define HSRP_OPCODE_ADVERTISE 3
static const value_string hsrp_opcode_vals[] = {
        {HSRP_OPCODE_HELLO,     "Hello"},
        {HSRP_OPCODE_COUP,      "Coup"},
        {HSRP_OPCODE_RESIGN,    "Resign"},
        {HSRP_OPCODE_ADVERTISE, "Advertise"},
   {0, NULL},
};
```

Ethereal

```
static void
dissect_hsrp(tvbuff_t *tvb, packet_info *pinfo,
   proto_tree *tree)
{
        guint8 opcode, state = 0;
[...]
        opcode = tvb_get_guint8(tvb, 1);
        if (check_col(pinfo->cinfo, COL_INFO)) {
                col_add_fstr(pinfo->cinfo, COL_INFO,
   "%s",
                             val_to_str(opcode,
   hsrp_opcode_vals, "Unknown"));
   }
[...]
        if (tree) {
[...]
                proto_tree_add_uint(hsrp_tree,
   hf_hsrp_opcode, tvb, offset, 1, opcode);
                offset++;
[...]
}
```



Ethereal

```
void proto_register_hsrp(void)
{
        static hf_register_info hf[] = {
[...]
                { &hf_hsrp_opcode,
                  { "Op Code", "hsrp.opcode",
                    FT_UINT8, BASE_DEC,
  VALS(hsrp_opcode_vals), 0x0,
                    "The type of message contained
  in this packet", HFILL }},
[...]
```

Ethereal

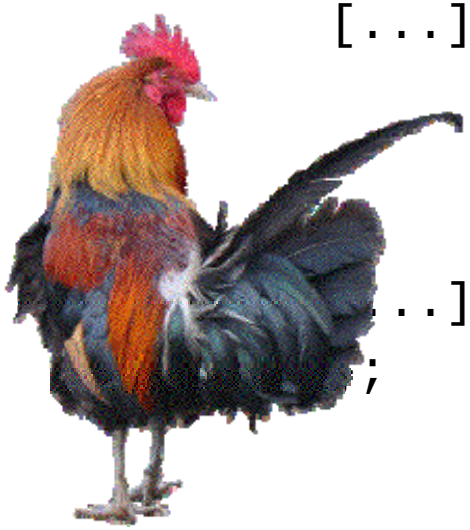
- 21x "opcode"
- 3x name of specific opcode
- 6x type of opcode (4 different names)
- 3x size of field (1 is implicit in offset++)

Yersinia

```
struct hsrp_data
{
[...]
    u_int8_t opcode;       /* Type of message */
[...]
}

#define HSRP_OPCODE      7
#define HSRP_DFL_TYPE        HSRP_TYPE_HELLO
```

Yersinia

```
static const struct tuple_type_desc hsrp_opcode[]
= {
    { HSRP_TYPE_HELLO, "(HELLO)" },
    { HSRP_TYPE_COUP, "(COUP)" },
    { HSRP_TYPE_RESIGN, "(RESIGN)" },
    { 0, NULL }
};

static struct position_fields hsrp_fields[] = {
[...]
        { HSRP_OPCODE, "Opcode",
          3, 13, 20, 2 , 0, 0,
          FIELD_HEX, FIELD_NORMAL, hsrp_opcode},
...]
;
```



Yersinia

```
struct hsrp_printable {
/* HSRP and Ethernet fields*/
[...]
    u_int8_t opcode[3];
[...]
```

Yersinia

```
int8_t
hsrp_send_packet(struct attacks *attacks)
{
[...]
    *aux = hsrp_data->opcode; aux++;
[...]
}
```

Yersinia

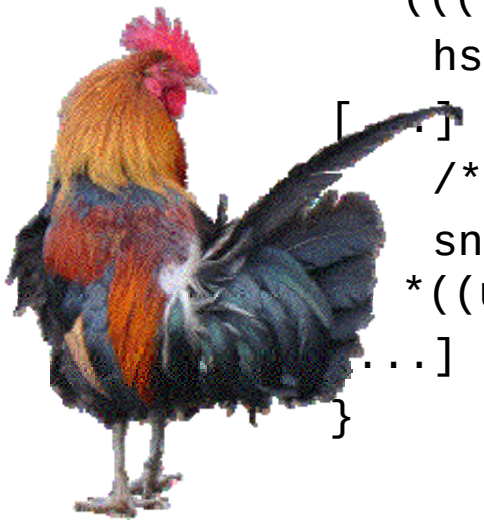
```
int8_t
hsrp_load_values(struct pcap_data *data, void *values)
{
[...]
    hsrp = (struct hsrp_data *)values;
    ether = (struct libnet_ethernet_hdr *) data->packet;
    ip_data = (u_char *) (data->packet + LIBNET_ETH_H);
    udp_data = (data->packet + LIBNET_ETH_H + (((*(data-
   >packet + LIBNET_ETH_H))&0x0F)*4));
    hsrp_data = udp_data + LIBNET_UDP_H;
[...]
    /* Opcode */
    hsrp->opcode = *((u_char *)hsrp_data+1);
[...]
```

Yersinia

```
int8_t
hsrp_init_attribs(struct term_node *node)
{
[...]
    /* HSRP stuff */
[...]
    hsrp_data->opcode  = HSRP_DFL_TYPE;
[...]
}
```

Yersinia

```
/*
 * Return formated strings of each HSRP field
 */
char **
hsrp_get_printable_packet(struct pcap_data *data)
{
[...]
    ether = (struct libnet_ethernet_hdr *) data->packet;
    ip_data = (char *) (data->packet + LIBNET_ETH_H);
    udp_data = (char *) (data->packet + LIBNET_ETH_H +
   (((*(data->packet + LIBNET_ETH_H))&0x0F)*4));
    hsrp_data = udp_data + LIBNET_UDP_H;
[...]
    /* Opcode */
    snprintf(field_values[HSRP_OPCODE], 3, "%02X",
   *((u_char *)hsrp_data+1));
[...]
}
```



Yersinia

```
char **
hsrp_get_printable_store(struct term_node *node)
{
[...]
    /* smac + dmac + sip + dip + sport + dport +
   ver + opcode + state + hello +
     * hold + priority + group + reserved + auth +
   vip + null = 17
     */
[...]
    /* Opcode */
    snprintf(field_values[HSRP_OPCODE], 3, "%02X",
   hsrp->opcode);
..]
```

Yersinia

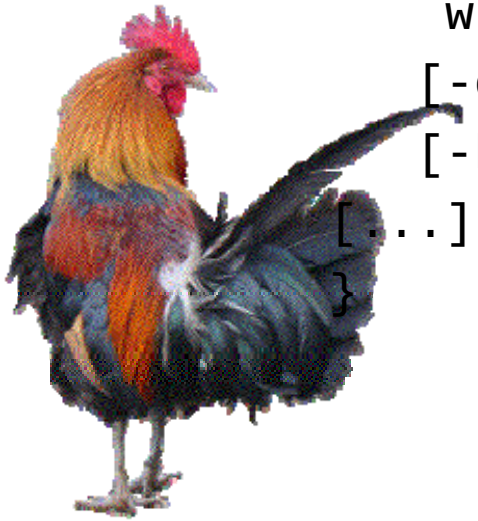
```
int8_t
hsrp_update_field(int8_t state, struct term_node
   *node, void *value)
{
[...]
    switch(state)
    {
[...]
        /* Op */
        case HSRP_OPCODE:
           hsrp_data->opcode = *(u_int8_t *)value;
        break;
[...]
}
```

Yersinia

```
int8_t
hsrp_update_data(int8_t state, int8_t key, int8_t
   position, struct term_node *node)
{
[...]
    switch(state)
    {
[...]
        /* Opcode */
        case HSRP_OPCODE:
            hsrp[7][position] = key;
            hsrp_data->opcode = strtoul(hsrp[7],
(char **)NULL, 16);
        break;
[...]
}
```

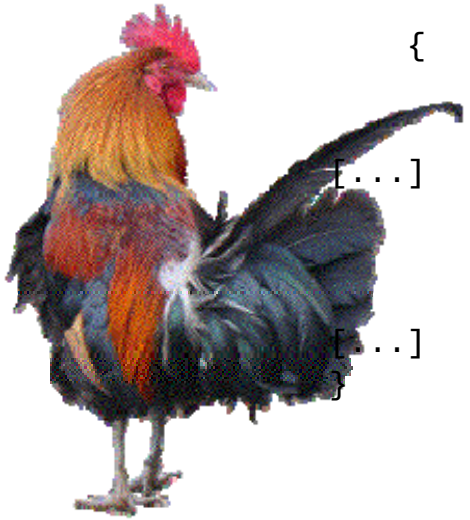
Yersinia

```
void
hsrp_help(void)
{
[...]
    write_log(2,"\nUsage: %s hsrp [-hM]
   [-i interface]\n", PACKAGE);
    write_log(2,"       [-smac hw_addr]”
   [-dmac hw_addr] [-sip addr] [-dip addr]”
   [-sport port] [-dport port]\n");
    write_log(2,"       [-version version]”
   [-opcode opcode] [-state state] “
   [-hello_time secs] [-hold_time secs]\n");
[...]
}
```



Yersinia

```
/*
 * HSRP parser...
 */
int8_t
hsrp_parser( struct term_node *node, int8_t argc, char **args, void
    *tmp_data)
{
[...]
    static struct option options[] =
    {
[...]
        { "opcode",       1, 0, 3 },
[...]

    while( (carac=getopt_long_only(argc,args,"h",
                    options, &opt_ind)) != EOF)
    {
        switch(carac)
        {
[...]
            case 3:
                hsrp_data->opcode = atoi(optarg);
            break;
[...]
    }
```



Yersinia

- 35x "opcode"
- 7x type (3 different, once as void*)
- length... who knows?

Scapy

```
class Field:
    islist=0
    def __init__(self, name, default, fmt="H"):
        self.name = name
        if fmt[0] in "@=<>!":
            self.fmt = fmt
        else:
            self.fmt = "!"+fmt
        self.default = self.any2i(None,default)
        self.sz = struct.calcsize(self.fmt)

[...]
    def addfield(self, pkt, s, val):
        return s+struct.pack(self.fmt,
    self.i2m(pkt,val))
    def getfield(self, pkt, s):
        return  s[self.sz:], self.m2i(pkt,
    struct.unpack(self.fmt, s[:self.sz])[0])
[...]
    (def for i2m, m2i, h2i, i2h, any2i, i2repr...)
```

Scapy

```
class ByteField(Field):
    def __init__(self, name, default):
        Field.__init__(self, name, default, "B")
```

Scapy

```
def mac2str(mac):
    return "".join(map(lambda x: chr(int(x,16)),
  mac.split(":")))

def str2mac(s):
    return ("%02x:"*6)[:-1] % tuple(map(ord, s))
```

Yersinia

```
/* Source MAC */
   snprintf(field_values[HSRP_SMAC], 18,
  "%02X:%02X:%02X:%02X:%02X:%02X",
      hsrp->mac_source[0], hsrp->mac_source[1],
      hsrp->mac_source[2], hsrp->mac_source[3],
      hsrp->mac_source[4], hsrp->mac_source[5]);
   /* Destination MAC */
   snprintf(field_values[HSRP_DMAC], 18,
  "%02X:%02X:%02X:%02X:%02X:%02X",
      hsrp->mac_dest[0], hsrp->mac_dest[1],
      hsrp->mac_dest[2], hsrp->mac_dest[3],
      hsrp->mac_dest[4], hsrp->mac_dest[5]);
```

Scapy

```
class MACField(Field):
    def __init__(self, name, default):
        Field.__init__(self, name, default, "6s")
    def i2m(self, pkt, x):
        if x is None:
            x="00:00:00:00:00:00"
        return mac2str(x)
    def m2i(self, pkt, x):
        return str2mac(x)
    def any2i(self, pkt, x):
        if type(x) is str and len(x) is 6:
            x = self.m2i(pkt, x)
        return x
    def i2repr(self, pkt, x):
        return self.i2h(pkt, x)
    def randval(self):
        return RandMAC()
```

Scapy

```
class DestMACField(MACField):
    def __init__(self, name):
        MACField.__init__(self, name, None)
    def i2h(self, pkt, x):
        if x is None:
            dstip = None
            if isinstance(pkt.payload, IPv6):
                dstip = pkt.payload.dst
            elif isinstance(pkt.payload, IP):
                dstip = pkt.payload.dst
            elif isinstance(pkt.payload, ARP):
                dstip = pkt.payload.pdst
            if isinstance(dstip, Gen):
                dstip = dstip.__iter__().next()
            if dstip is not None:
                if isinstance(pkt.payload, IPv6):
                    x = getmacbyip6(dstip)
                else:
                    x = getmacbyip(dstip)
            if x is None:
                x = "ff:ff:ff:ff:ff:ff"
                warning("Mac address to reach %s not found\n"%dstip)
        return MACField.i2h(self, pkt, x)
    def i2m(self, pkt, x):
        return MACField.i2m(self, pkt, self.i2h(pkt, x))
```

Scapy

```
def getmacbyip(ip):
        iff,a,gw = conf.route.route(ip)
        if ( (iff == "lo") or (ip == conf.route.get_if_bcast(iff)) ):
            return "ff:ff:ff:ff:ff:ff"
        if gw != "0.0.0.0":
            ip = gw

        if arp_cache.has_key(ip):
            mac, timeout = arp_cache[ip]
            if timeout and (time.time()-timeout < ARPTIMEOUT):
                return mac

        res = srp1(Ether(dst=ETHER_BROADCAST)/ARP(op="who-has", pdst=ip),
                   type=ETH_P_ARP,
                   iface = iff,
                   timeout=2,
                   verbose=0,
                   nofilter=1)
        if res is not None:
            mac = res.payload.hwsrc
            arp_cache[ip] = (mac,time.time())
            return mac
        return None
```

Scapy

```
class Packet(Gen):
    def __init__(self, _pkt="",
     _internal=0, **fields):
    def dissection_done(self,pkt):
    def post_dissection(self, pkt):
    def get_field(self, fld):
    def add_payload(self, payload):
    def remove_payload(self):
    def add_underlayer(self, underlayer):
    def remove_underlayer(self,other):
    def copy(self):
    def __getattr__(self, attr):
    def __setattr__(self, attr, val):
    def __delattr__(self, attr):
    def __repr__(self):
    def __str__(self):
    def __div__(self, other):
    def __rdiv__(self, other):
    def __len__(self):
    def do_build(self):
    def post_build(self, pkt):
    def build(self,internal=0):
    def do_build_ps(self):
    def build_ps(self,internal=0):
    def psdump(self, filename=None,
     **kargs):
    def pdfdump(self, filename=None,
     **kargs):
    def canvas_dump(self, layer_shift=0,
     rebuild=1):
    def extract_padding(self, s):
```

```
def post_dissect(self, s):
def do_dissect(self, s):
def do_dissect_payload(self, s):
def dissect(self, s):
def guess_payload_class(self, payload):
def default_payload_class(self, payload):
def hide_defaults(self):
def __iter__(self):
def send(self, s, slp=0):
def __gt__(self, other):
def __lt__(self, other):
def hashret(self):
def answers(self, other):
def haslayer(self, cls):
def haslayer_str(self, cls):
def getlayer(self, cls, nb=1):
def __getitem__(self, cls):
def __contains__(self, cls):
def display(self,*args,**kargs):  #
 Deprecated. Use show()
def show(self, lvl=0):
def show2(self):
def sprintf(self, fmt, relax=1):
def mysummary(self):
def summary(self, intern=0):
def lastlayer(self,layer=None):
def decode_payload_as(self,cls):
def libnet(self):
def command(self):
```

Scapy

```
class Packet(Gen):
    fields_desc = []

    aliastypes = []
    overload_fields = {}
[...]
    def __getattr__(self, attr):
        if self.initialized:
            fld = self.fieldtype.get(attr)
            if fld is None:
                i2h = lambda x,y: y
            else:
                i2h = fld.i2h
            for f in ["fields", "overloaded_fields",
"default_fields"]:
                fields = self.__dict__[f]
                if fields.has_key(attr):
                    return i2h(self, fields[attr] )
            return getattr(self.payload, attr)
        raise AttributeError(attr)
```

Scapy

```
class Packet(Gen):
[...]
    def do_build(self):
        p=""
        for f in self.fields_desc:
            p = f.addfield(self, p, self.__getattr__(f))
        pkt = p+self.payload.build(internal=1)
        return pkt

    def post_build(self, pkt):
        return pkt

    def build(self,internal=0):
        p = self.post_build(self.do_build())
        if not internal:
            pkt = self
            while pkt.haslayer(Padding):
                pkt = pkt.getlayer(Padding)
                p += pkt.load
                pkt = pkt.payload
        return p
```

Scapy

```
class Packet(Gen):
[...]
    def show(self, lvl=0):
        ct = conf.color_theme
        print "%s %s %s" % (ct.punct("###["),
                            ct.layer_name(self.name),
                            ct.punct("]###"))
        for f in self.fields_desc:
            if isinstance(f, Emph):
                ncol = ct.emph_field_name
                vcol = ct.emph_field_value
            else:
                ncol = ct.field_name
                vcol = ct.field_value
            print "  %-10s%s %s" % (ncol(f.name),
                                    ct.punct("="),
                       vcol(f.i2repr(self,self.__getattr__(f))))
        self.payload.display(lvl+1)
```

Scapy

```
layer_bonds = [ ( Dot3,   LLC,      { } ),
                ( GPRS,   IP,       { } ),
                ( PrismHeader, Dot11, { }),
                ( Dot11,  LLC,      { "type" : 2 } ),
                ( PPP,    IP,       { "proto" : 0x0021 } ),
                ( Ether,  LLC,      { "type" : 0x007a } ),
                ( Ether,  Dot1Q,    { "type" : 0x8100 } ),
                ( Ether,  Ether,    { "type" : 0x0001 } ),
                ( Ether,  ARP,      { "type" : 0x0806 } ),
                ( Ether,  IP,       { "type" : 0x0800 } ),
                ( Ether,  EAPOL,    { "type" : 0x888e } ),
```

Scapy

```
>>>  a=Ether()/IP(dst="www.slashdot.org")/TCP()/"GET /index.html HTTP/1.0
    \n\n"
>>>  hexdump(a)
00 02 15 37 A2 44 00 AE F3 52 AA D1 08 00 45 00  ...7.D...R....E.
00 43 00 01 00 00 40 06 78 3C C0 A8 05 15 42 23  .C....@.x<....B#
FA 97 00 14 00 50 00 00 00 00 00 00 00 00 50 02  .....P........P.
20 00 BB 39 00 00 47 45 54 20 2F 69 6E 64 65 78   ..9..GET /index
2E 68 74 6D 6C 20 48 54 54 50 2F 31 2E 30 20 0A  .html HTTP/1.0 .
0A                                               .
>>> b=str(a)
>>> b
'\x00\x02\x157\xa2D\x00\xae\xf3R\xaa\xd1\x08\x00E\x00\x00C\x00\x01\x00\x00@
    \x06x<\xc0\xa8\x05\x15B#\xfa\x97\x00\x14\x00P\x00\x00\x00\x00\x00\x00\x
    00\x00P\x02\x00\xbb9\x00\x00GET /index.html HTTP/1.0 \n\n'
```

Scapy

```
send(IP(dst="1.2.3.4")/ICMP())
```

Scapy

```
def fuzz(p):
    p = q = p.copy()
    while not isinstance(q, NoPayload):
        for f in q.fields_desc:
            if f.default is not None:
                rnd = f.randval()
                if rnd is not None:
                    q.default_fields[f] = rnd
        q = q.payload
    return p
```

Scapy

```
send(IP(dst="target")/fuzz(UDP()/NTP(version=4)),loop=1)
```

Scapy

```
>>> sr(IP(dst="192.168.8.1")/TCP(dport=[21,22,23]))
Received 6 packets, got 3 answers, remaining 0 packets
(<Results: UDP:0 TCP:3 ICMP:0 Other:0>, <Unanswered: UDP:0 TCP:0 ICMP:0
    Other:0>)
>>> ans,unans=_
>>> ans.summary()
IP / TCP 192.168.8.14:20 > 192.168.8.1:21 S ==> Ether / IP / TCP
    192.168.8.1:21 > 192.168.8.14:20 RA / Padding
IP / TCP 192.168.8.14:20 > 192.168.8.1:22 S ==> Ether / IP / TCP
    192.168.8.1:22 > 192.168.8.14:20 RA / Padding
IP / TCP 192.168.8.14:20 > 192.168.8.1:23 S ==> Ether / IP / TCP
    192.168.8.1:23 > 192.168.8.14:20 RA / Padding
```

Internationalisierungsbestrebungen

Internationalisierung Marke Real:

ppvcfg.cpp:

```
   // Report an error
   char pError[512];
   sprintf(pError, pMessageTable[CE_AMBIGUOUS_AUTH_MODE],
       (const char*)(pRule->m_Name));
```

Fehlt was ?

OpenSSL num num:

```
void AES_cfb8_encrypt(
        const unsigned char *in, unsigned char *out,
        const unsigned long length, const AES_KEY *key,
        unsigned char *ivec, int *num, const int enc)
    {
    unsigned int n;

    assert(in && out && key && ivec && num);
    assert(*num == 0);

    for(n=0 ; n < length ; ++n)
        AES_cfbr_encrypt_block(
           &in[n],&out[n],8,key,ivec,enc);
    }
```

WTF ?

```
private int ScanGCW (void * pGlobals, unsigned int cbBitArray, byte
   *pByteArray)
{
   unsigned int  iWord, iBit;
   unsigned int  iCol;
   const unsigned int  UNALIGNED *pBitArray;
   static const unsigned int Masks[16] =
        {0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080,
         0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4000, 0x8000};
   iCol = 0;
   pBitArray = (unsigned int *)pByteArray;

   for (iWord = 0; iWord < (cbBitArray / 2); iWord++) {
      for (iBit = 0; iBit < 16; iBit++) {
         if ((*pBitArray & Masks[iBit]) != 0)
            pCurrentFile->colSize[iCol].isStandardWidth= TRUE;
         iCol++;
      }
      pBitArray++;
   }
   return (EX_errSuccess);
}
```

Neue Trends

Hier der neueste IT-Trend aus den Vereinigten Staaten, Multicasting:

```
*pcbRead = (ULONG) ( (BYTE*)(WCHAR*)pUnicode - (BYTE*)pv );
Assert(*pcbRead == 0);
```

Neue Trends

Ein weiterer Trend, IGC (implicit garbage collection):

```
inline void MemFreeAllPages (void * pGlobals) { }
```

Debuggen? Niemals.

Boomerang pentiumdecoder.cpp: ***62083*** lines

```
case 3:
  MATCH_w_8_8 = getByte(1 + MATCH_p);
  switch((MATCH_w_8_8 >> 6 & 0x3) /* mod at 8 */) {
      case 0: switch((MATCH_w_8_8 & 0x7) /* r_m at 8 */) {
            case 0: case 1: case 2: case 3: case 6: case 7:
              goto MATCH_label_c797; break;
            case 4:
              MATCH_w_8_16 = getByte(2 + MATCH_p);
              if ((MATCH_w_8_16 & 0x7)
                      /* base at 16 */ == 5 &&
                (0 <= (MATCH_w_8_16 >> 3 & 0x7)
                      /* index at 16 */ &&
                (MATCH_w_8_16 >> 3 & 0x7)
                      /* index at 16 */ < 8))
                goto MATCH_label_c799;  /*opt-block+*/
              else
                goto MATCH_label_c798;  /*opt-block+*/

              break;
            case 5:
```

REAL's Base 64 Decoder [1/4]

```
#define XX 127
/*
 * Table for decoding base64
 */
static const char Index64[256] = { /* Flawfinder: ignore */
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,62,XX,XX, XX,XX,XX,XX, XX,XX,63,62, XX,XX,XX,63,
    52,53,54,55, 56,57,58,59, 60,61,XX,XX, XX,XX,XX,XX,
    XX, 0, 1, 2,  3, 4, 5, 6,  7, 8, 9,10, 11,12,13,14,
    15,16,17,18, 19,20,21,22, 23,24,25,XX, XX,XX,XX,XX,
    XX,26,27,28, 29,30,31,32, 33,34,35,36, 37,38,39,40,
    41,42,43,44, 45,46,47,48, 49,50,51,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX,
    XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX, XX,XX,XX,XX
};
#define CHAR64(c)  (Index64[(unsigned char)(c)])
```

REAL‘s Base 64 Decoder [2/4]

```
INT32 BinFrom64(const char* pInBuf, INT32 len, BYTE* pOutBuf)
{
    int c1, c2, c3, c4;
    INT32 offset = 0;
    INT32 outOffset = 0;

    while(offset < len)
    {
    c1 = pInBuf[offset++];
    if(c1 != '=' && CHAR64(c1) == XX)
    {
        continue;
    }
    else if (offset == len)
    {
        /* BAD data */
        return -1;
    }
    do
    {
        c2 = pInBuf[offset++];
    }
    while(offset < len && c2 != '=' && CHAR64(c2) == XX);
    if (offset == len)
    {
        /* BAD data */ return -1;
    }
```

REAL‘s Base 64 Decoder [3/4]

```
do
{
    c2 = pInBuf[offset++];
}
while(offset < len && c2 != '=' && CHAR64(c2) == XX);
if (offset == len)
{
    /* BAD data */
    return -1;
}
    do
{
    c3 = pInBuf[offset++];
}
while(offset < len && c3 != '=' && CHAR64(c3) == XX);
if (offset == len)
{
    /* BAD data */
    return -1;
}
do
{
    c4 = pInBuf[offset++];
}
while(offset < len && c4 != '=' && CHAR64(c4) == XX);
```

REAL‘s Base 64 Decoder [4/4]

```
    if (offset == len && c4 != '=' && CHAR64(c4) == XX)
    {
        /* BAD data */ return -1;
    }
    if(c1 == '=' || c2 == '=')
    {   continue;  }
    c1 = CHAR64(c1);
    c2 = CHAR64(c2);
    pOutBuf[outOffset++]  = ((c1 << 2) | ((c2 & 0x30) >> 4));
    if(c3 == '=')
    {   continue;  }
    else
    {
        c3 = CHAR64(c3);
        pOutBuf[outOffset++]  = (((c2 & 0x0f) << 4) | ((c3 & 0x3c) >> 2));
        if(c4 == '=')
        {           continue;     }
        else
        {
          c4 = CHAR64(c4);
          pOutBuf[outOffset++] = (((c3 & 0x03) << 6) | c4);
        }
    }
    }

    return outOffset;
}
```

Schlusswort

Lest mehr Code.