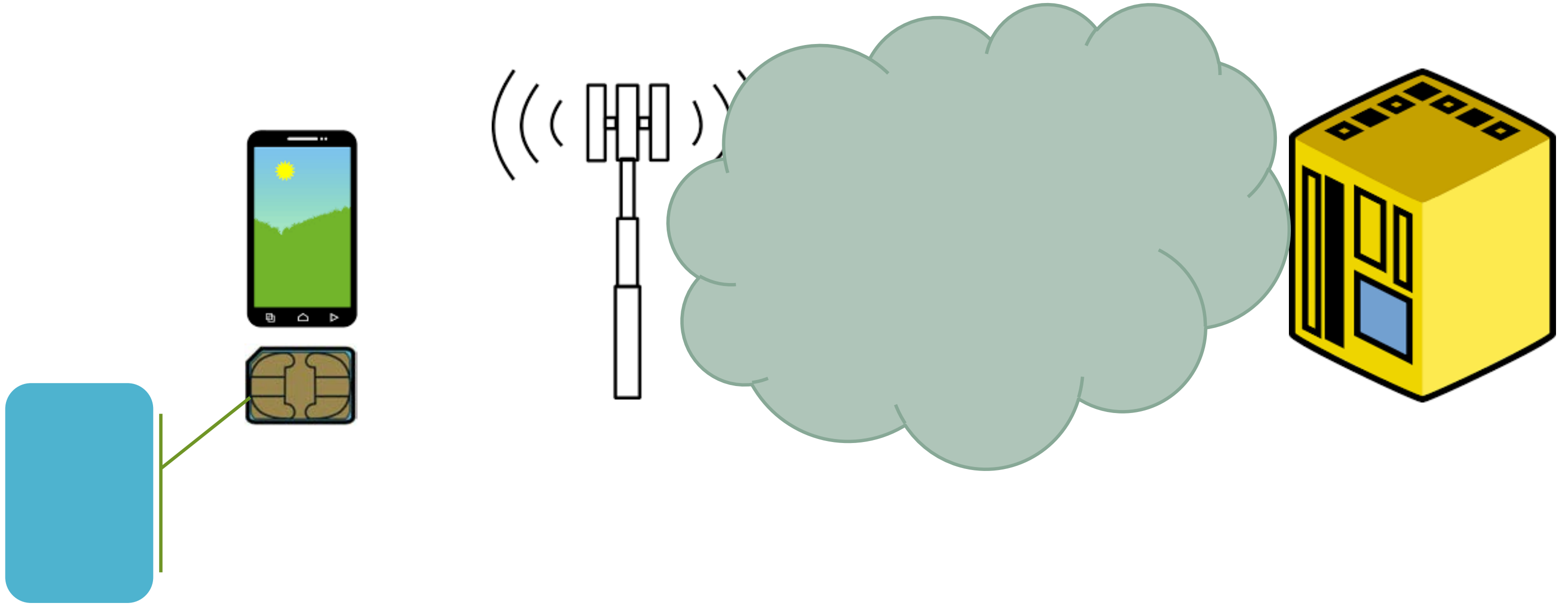
MOBILE INTERCONNECT THREATS

How next-gen products may be already outdated

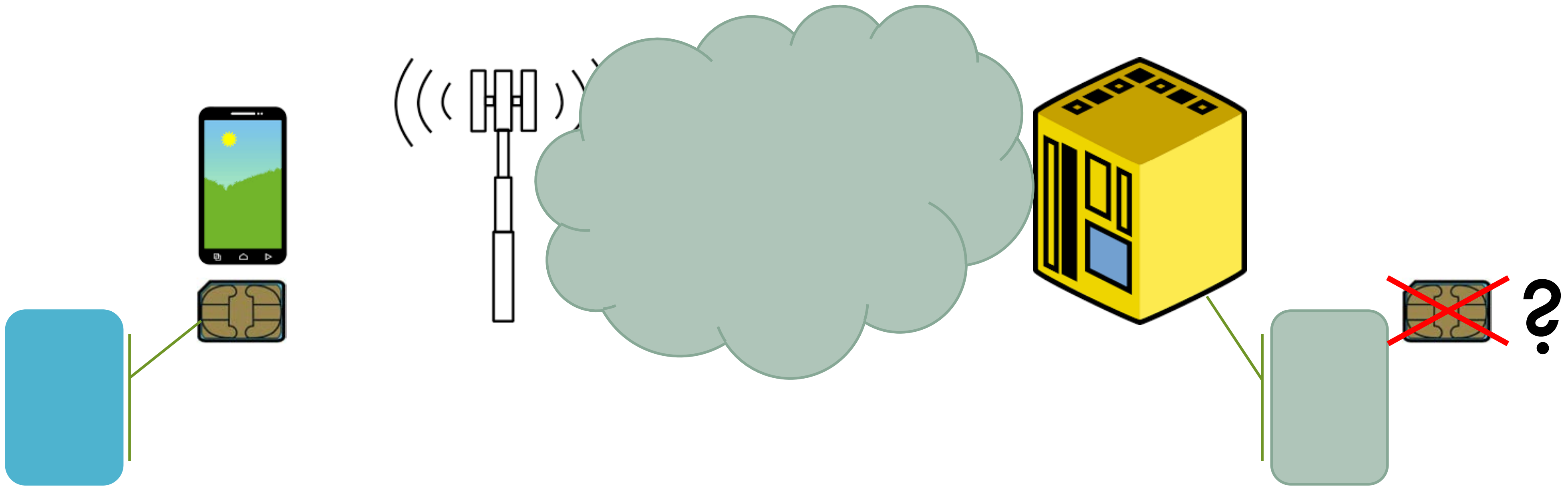
AGENDA

- Brief introduction to mobile interconnect, threats and solutions
- Unwinding the SIGTRAN stack to discover bugs
- Key takeaways

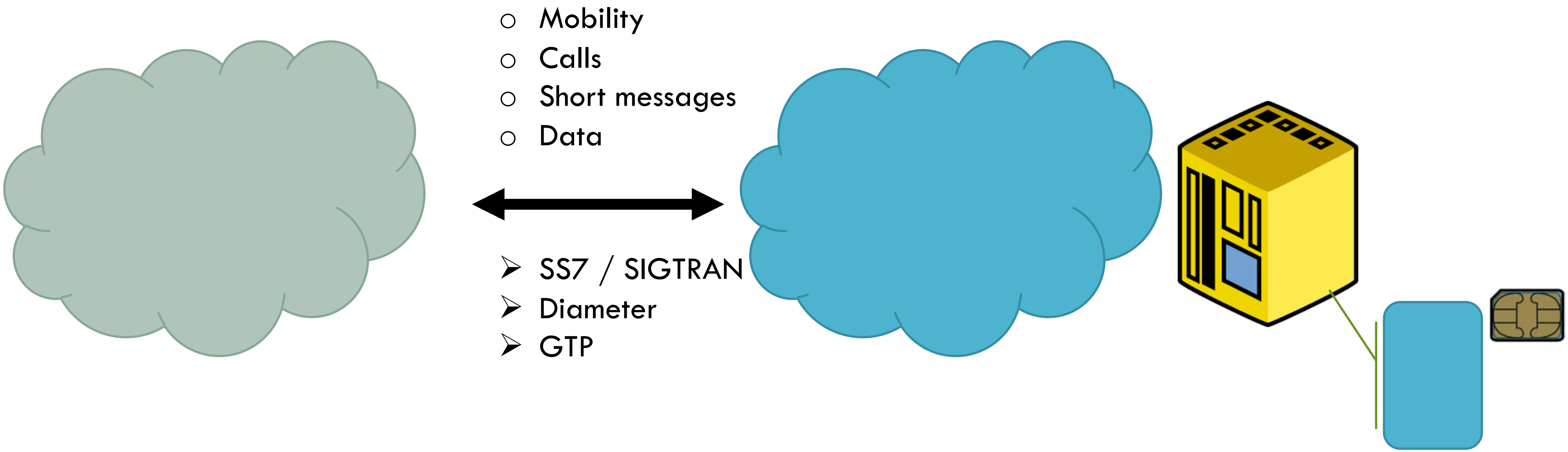
INTERCONNECT 101



INTERCONNECT 102



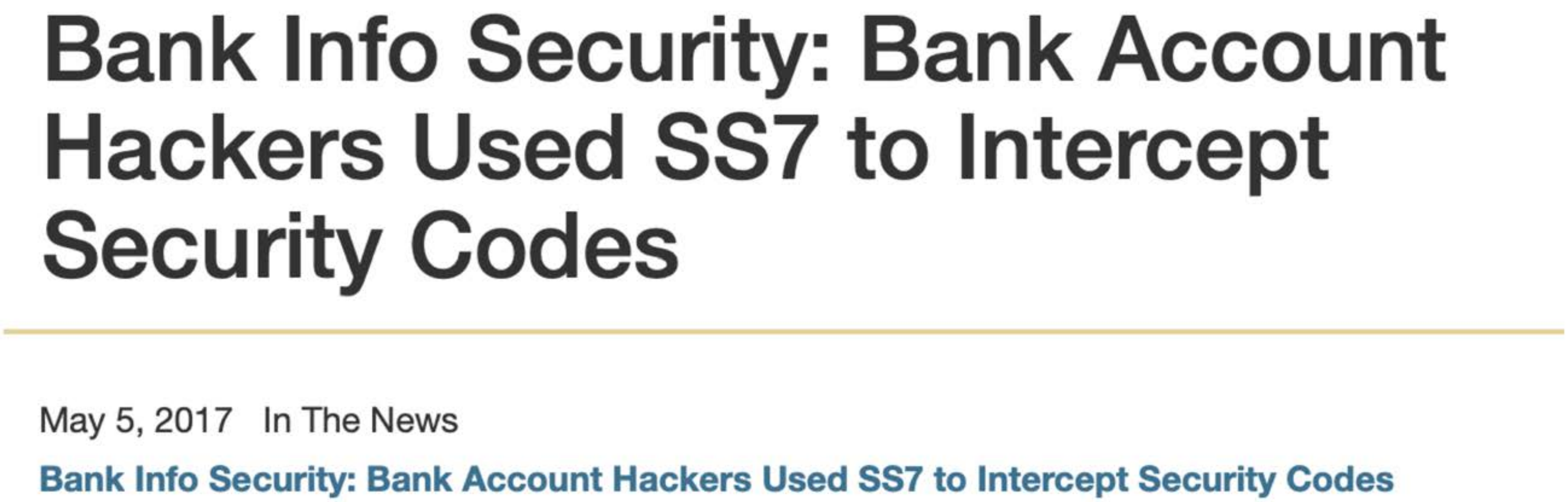
INTERCONNECT 103



- ○ Mobility
- ○ Calls
- ○ Short messages
- ○ Data

- ➢ SS7 / SIGTRAN
- ➢ Diameter
- ➢ GTP

MOBILE INTERCONNECT THREATS: A REALITY

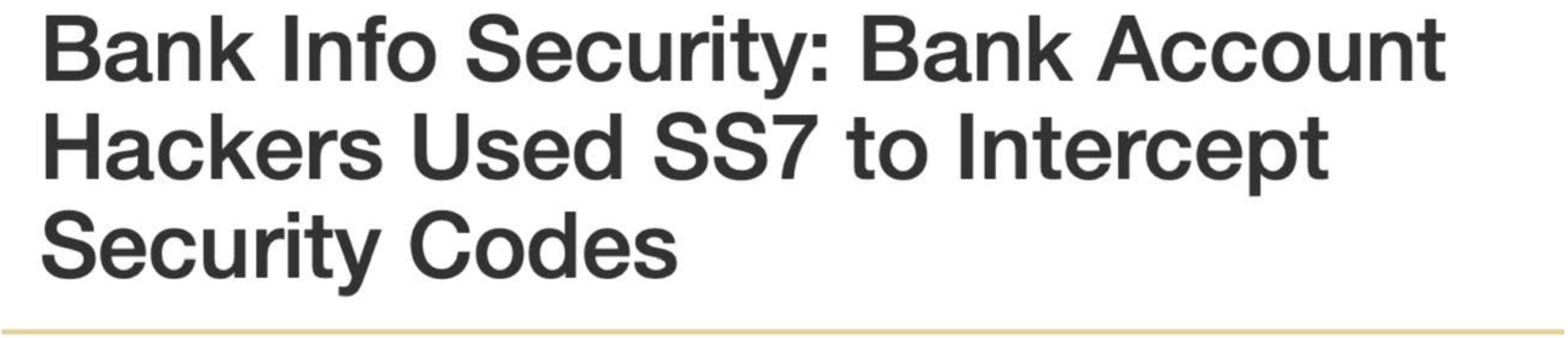


Bank Info Security: Bank Account Hackers Used SS7 to Intercept Security Codes

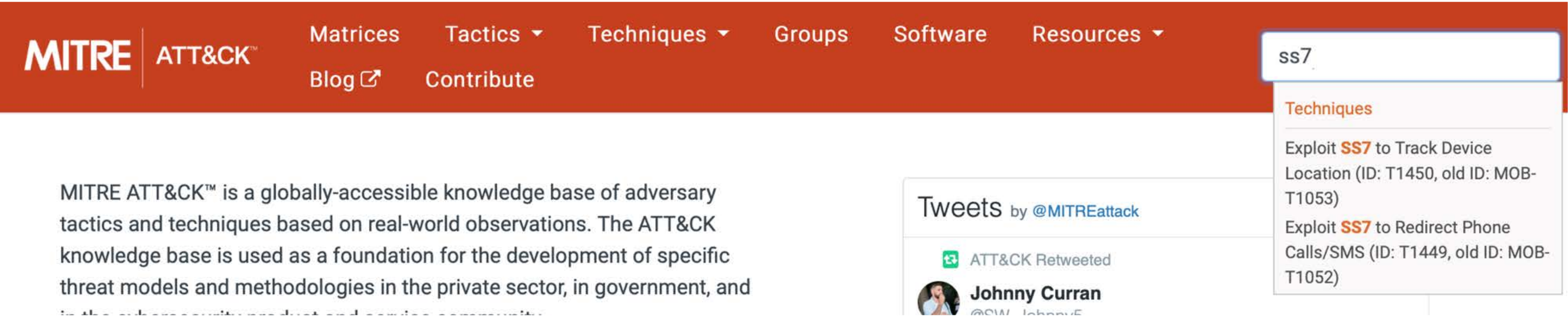
May 5, 2017 In The News

Bank Info Security: Bank Account Hackers Used SS7 to Intercept Security Codes

MOBILE INTERCONNECT THREATS: A REALITY



Bank Info Security: Bank Account Hackers Used SS7 to Intercept Security Codes



Techniques

Exploit SS7 to Track Device Location (ID: T1450, old ID: MOB-T1053)

Exploit SS7 to Redirect Phone Calls/SMS (ID: T1449, old ID: MOB-T1052)

MITRE ATT&CK™ is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and

Tweets by @MITREattack

ATT&CK Retweeted

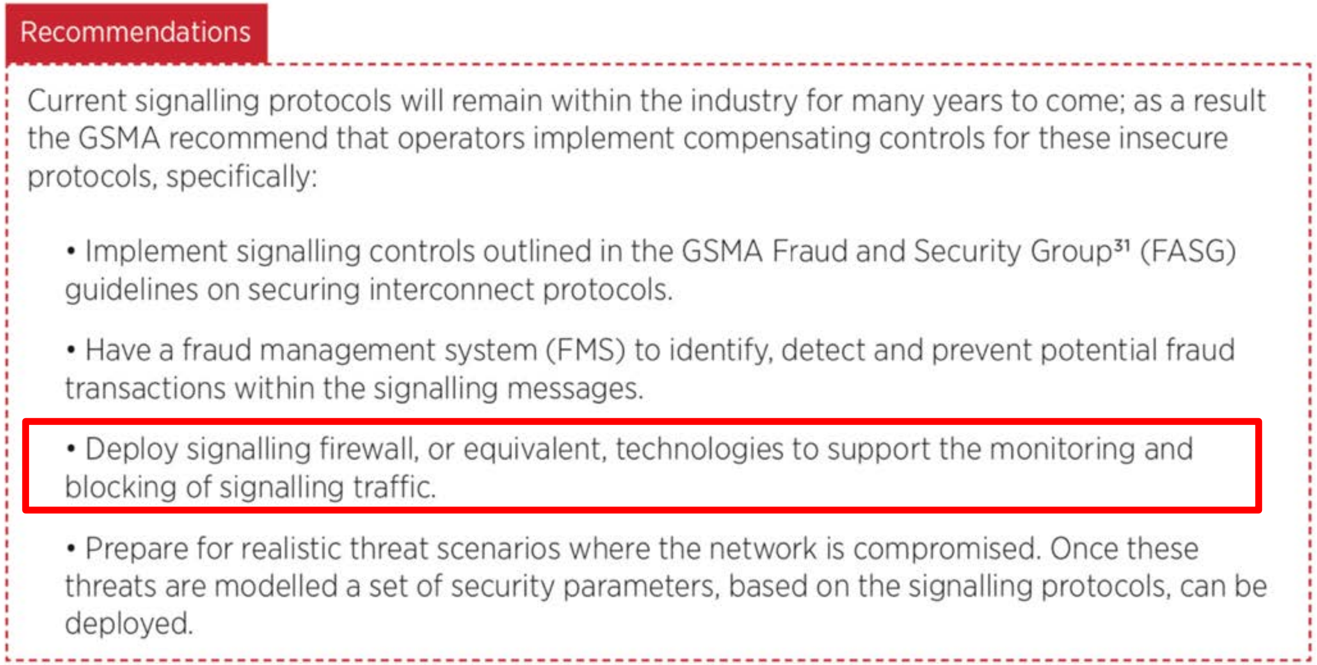
Johnny Curran

A MOVE TO DEFENSE

GSMA

Mobile Telecommunications Security Threat Landscape

January 2019

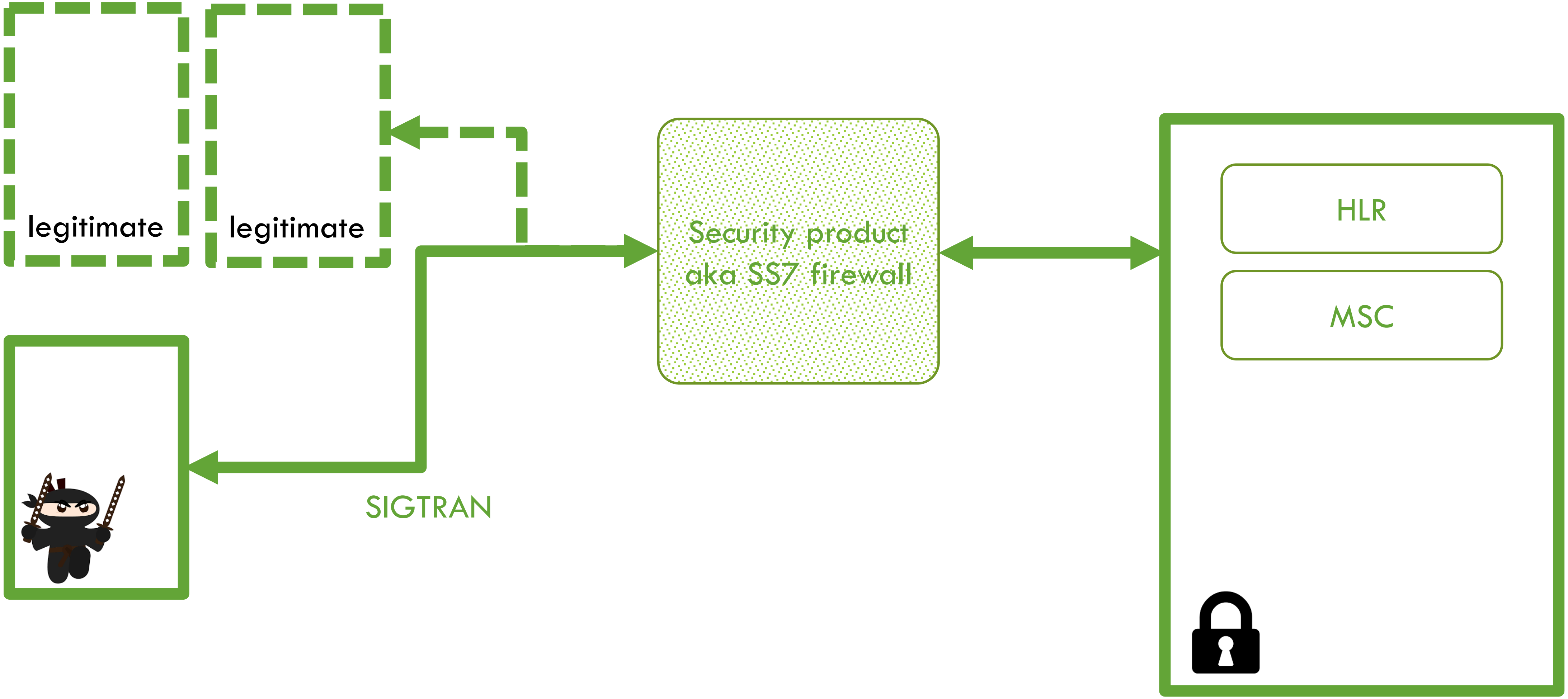


Recommendations

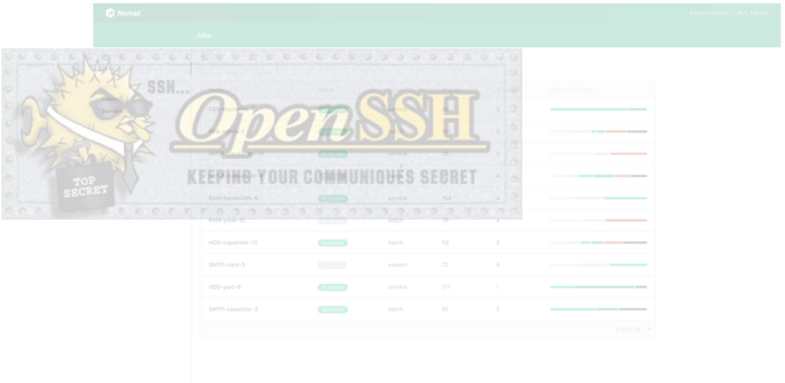
Current signalling protocols will remain within the industry for many years to come; as a result the GSMA recommend that operators implement compensating controls for these insecure protocols, specifically:

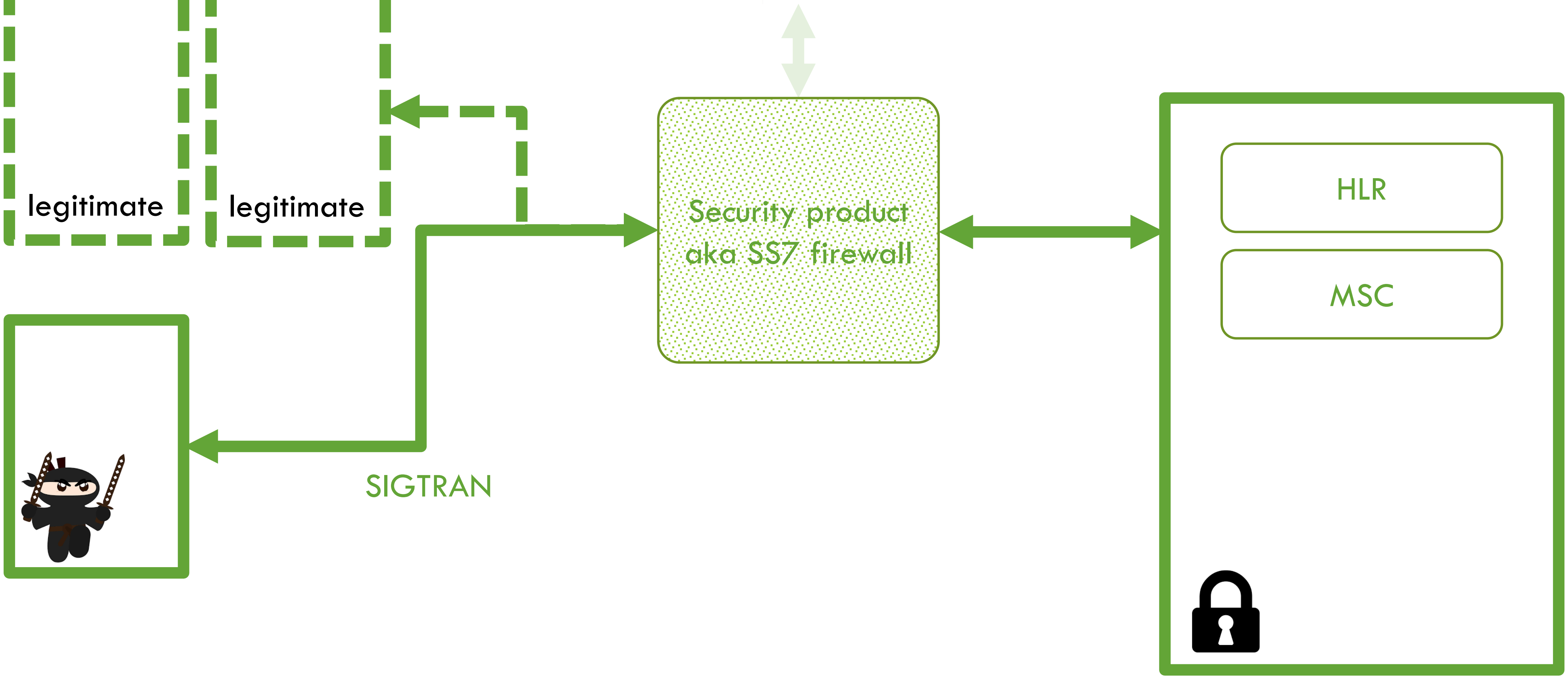
- Implement signalling controls outlined in the GSMA Fraud and Security Group[31] (FASG) guidelines on securing interconnect protocols.
- Have a fraud management system (FMS) to identify, detect and prevent potential fraud transactions within the signalling messages.
- Deploy signalling firewall, or equivalent, technologies to support the monitoring and blocking of signalling traffic.
- Prepare for realistic threat scenarios where the network is compromised. Once these threats are modelled a set of security parameters, based on the signalling protocols, can be deployed.

THREAT MODEL



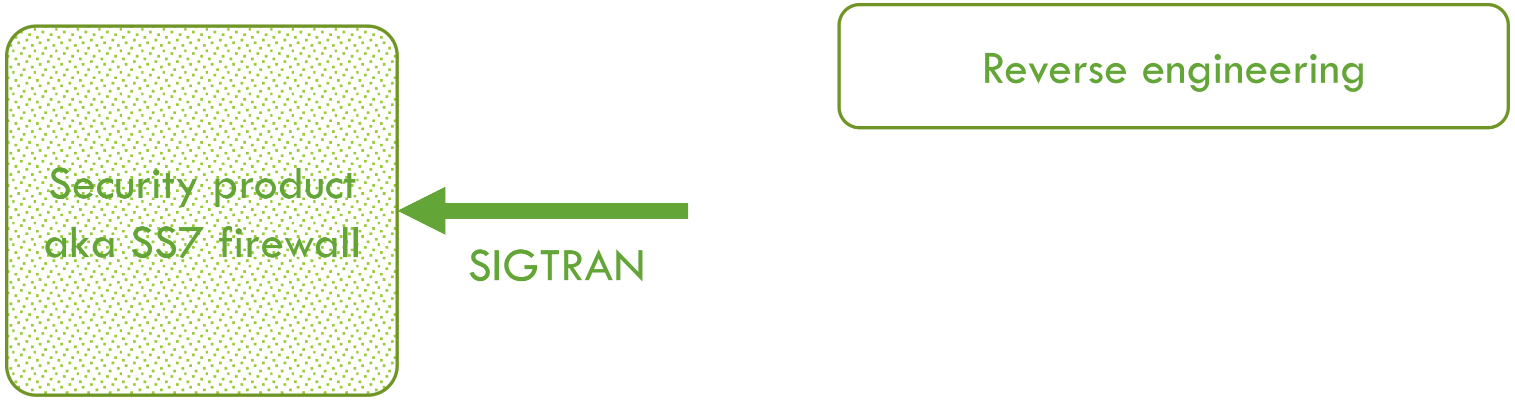
THREAT MODEL



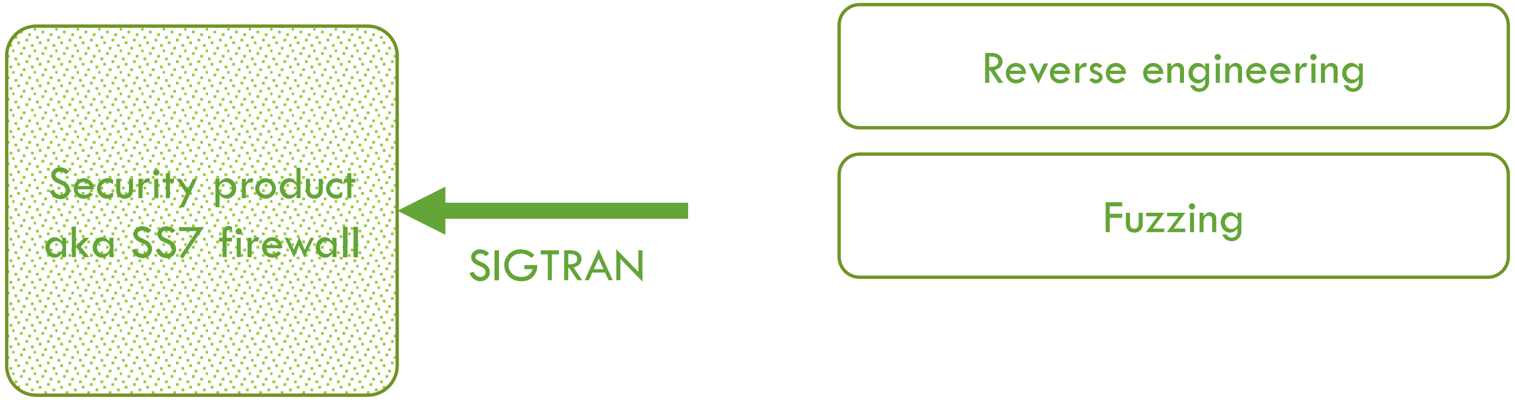


Security product aka SS7 firewall
SIGTRAN

OUR CONTRIBUTION



OUR CONTRIBUTION



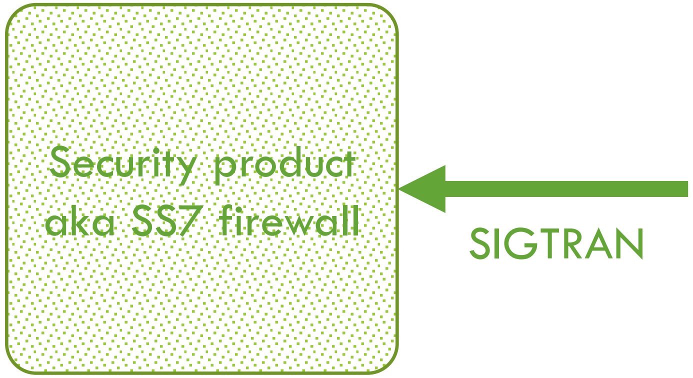
Security product aka SS7 firewall

SIGTRAN

Reverse engineering

Fuzzing

OUR CONTRIBUTION



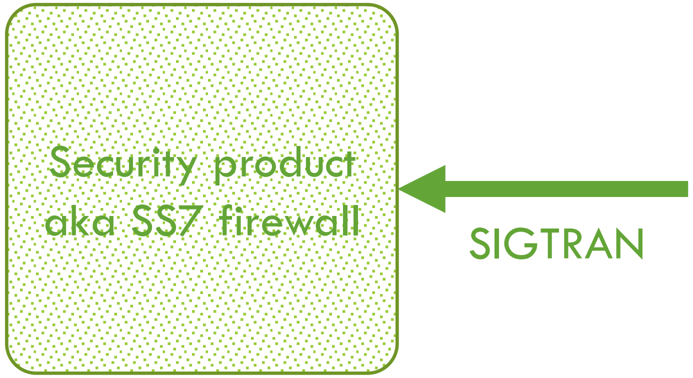
Security product aka SS7 firewall

SIGTRAN

- Reverse engineering
- Fuzzing
- Exploit development

OUR CONTRIBUTION

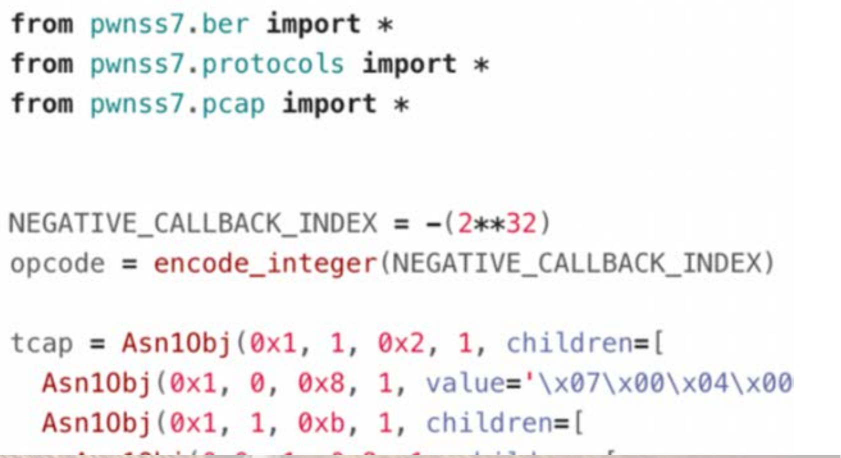
OUR CONTRIBUTION



Reverse engineering

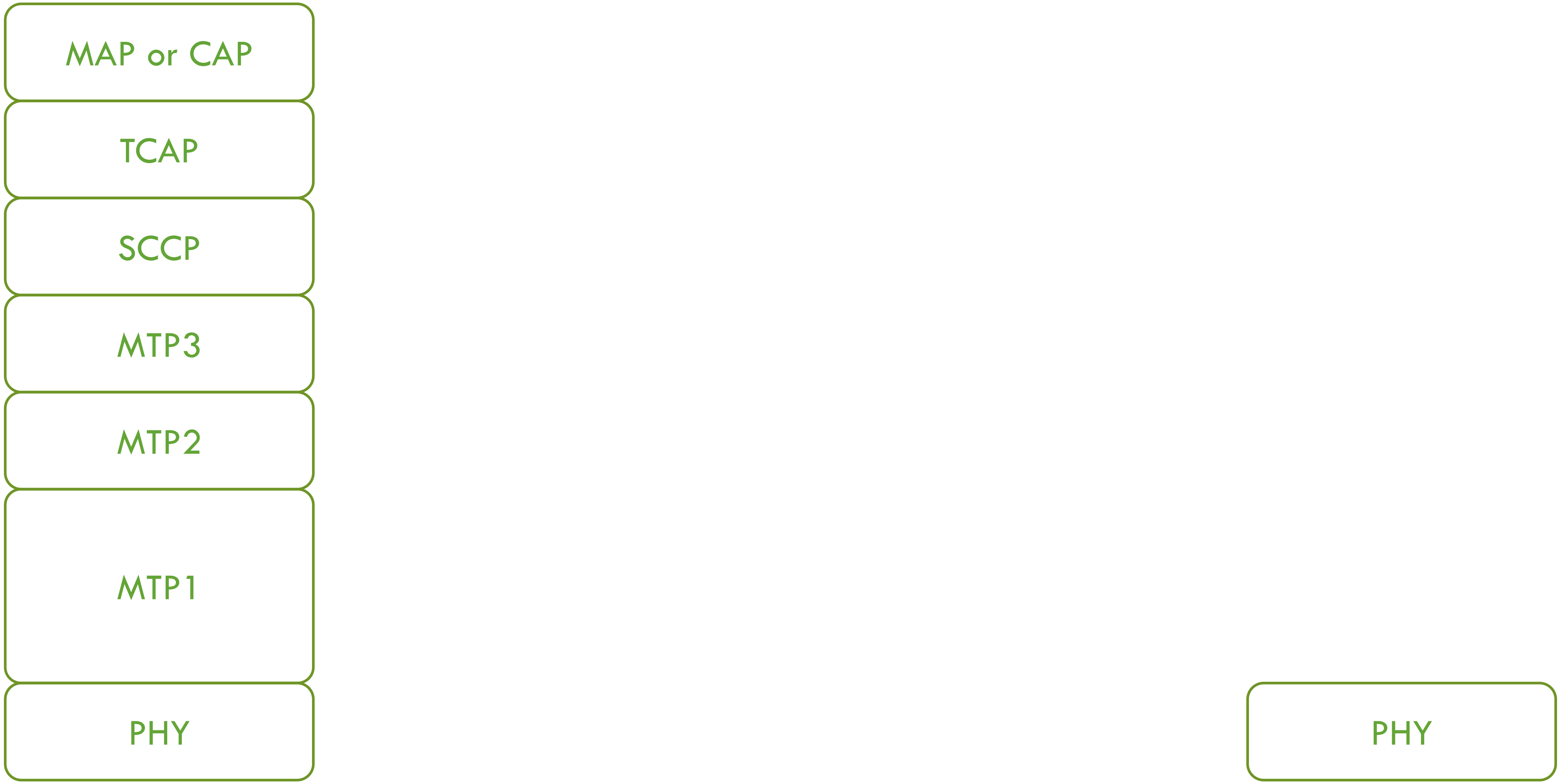
Fuzzing

Exploit development

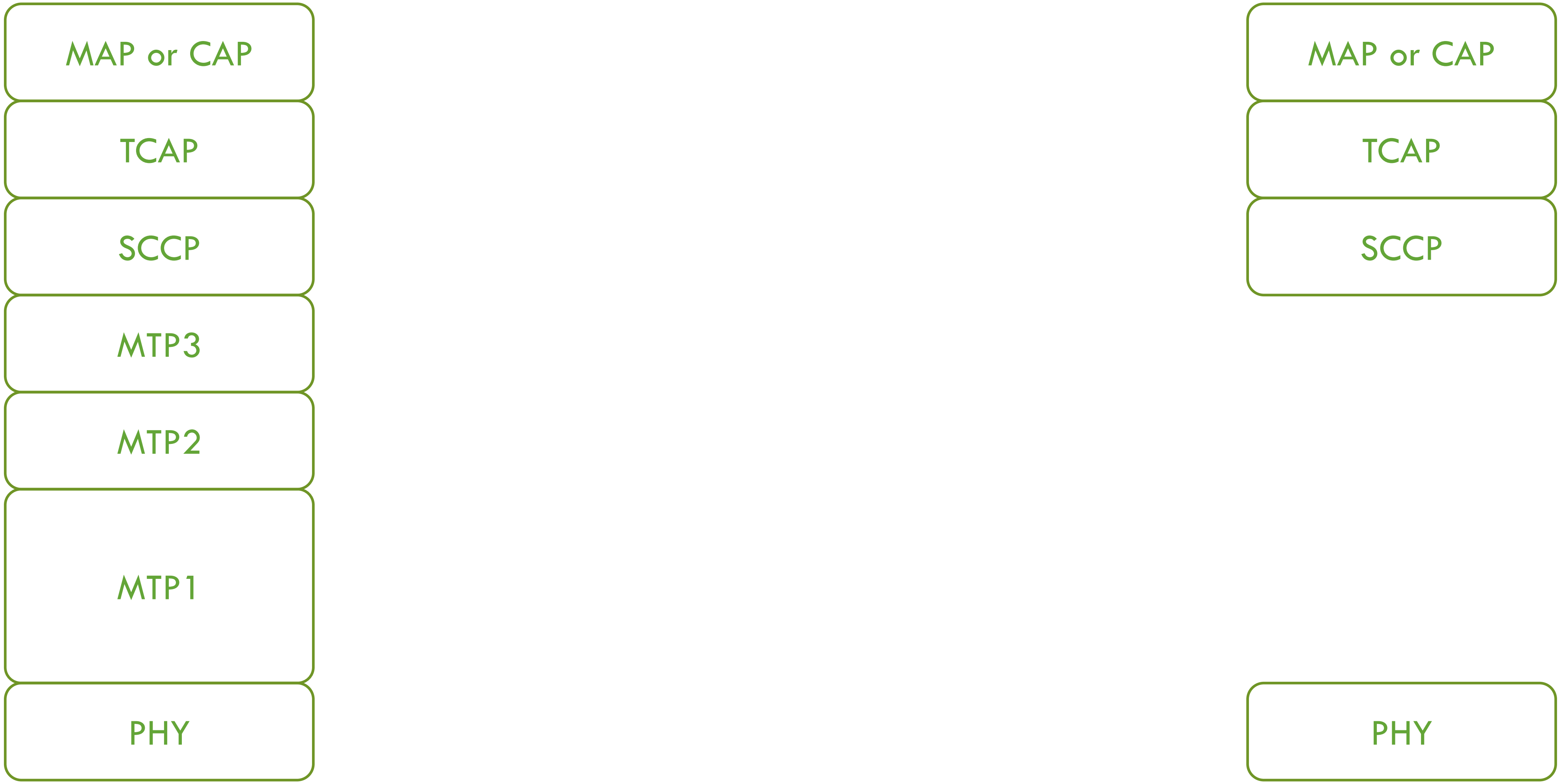


```
from pwnss7.ber import *
from pwnss7.protocols import *
from pwnss7.pcap import *


NEGATIVE_CALLBACK_INDEX = -(2**32)
opcode = encode_integer(NEGATIVE_CALLBACK_INDEX)

tcap = Asn1Obj(0x1, 1, 0x2, 1, children=[
  Asn1Obj(0x1, 0, 0x8, 1, value='\x07\x00\x04\x00
  Asn1Obj(0x1, 1, 0xb, 1, children=[
```

SS7 ON TOP OF IP TRANSPORT = SIGTRAN



SS7 ON TOP OF IP TRANSPORT = SIGTRAN



MAP or CAP

TCAP

SCCP

MTP3

MTP2

MTP1

PHY

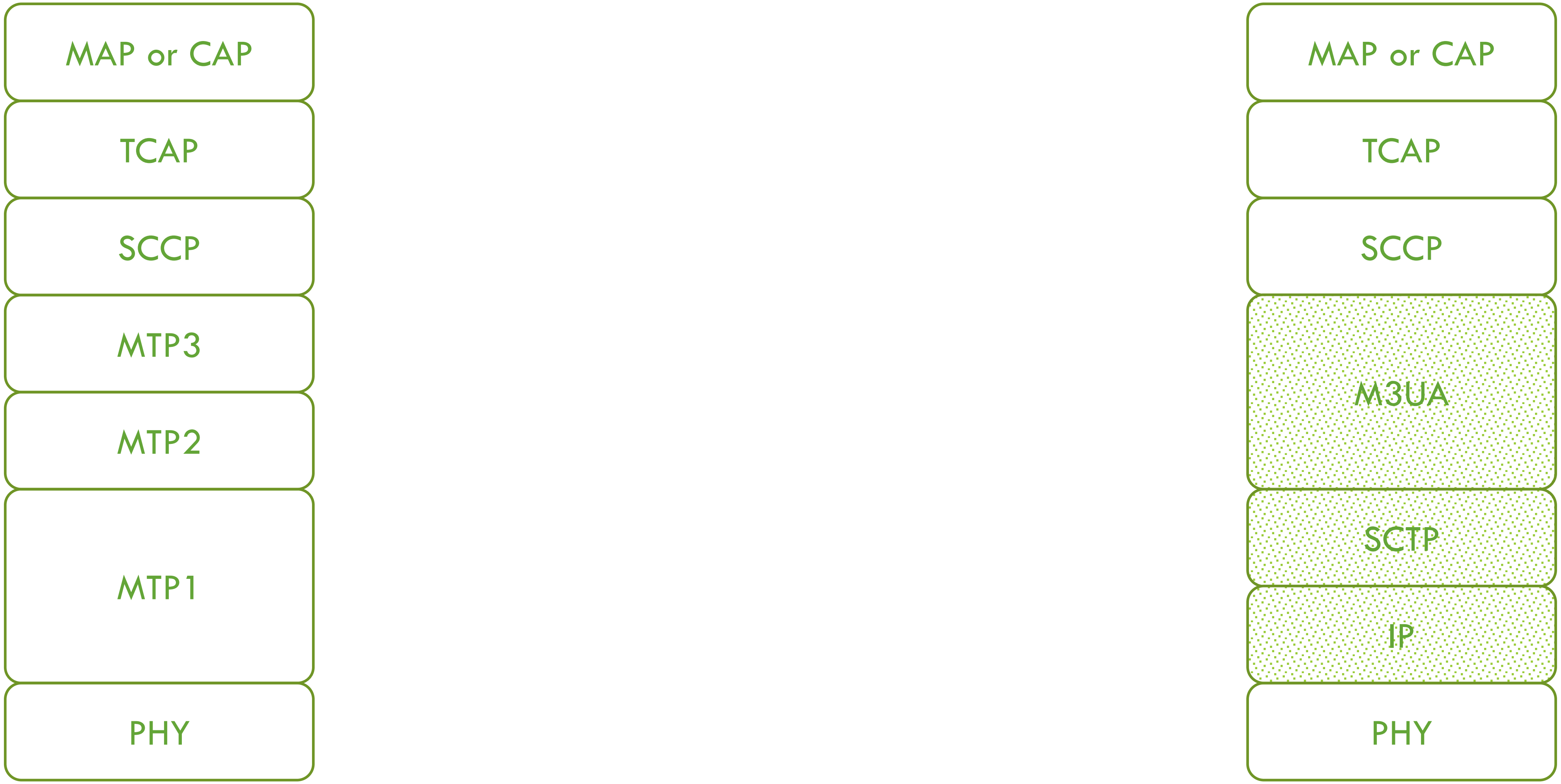
MAP or CAP

TCAP

SCCP

PHY

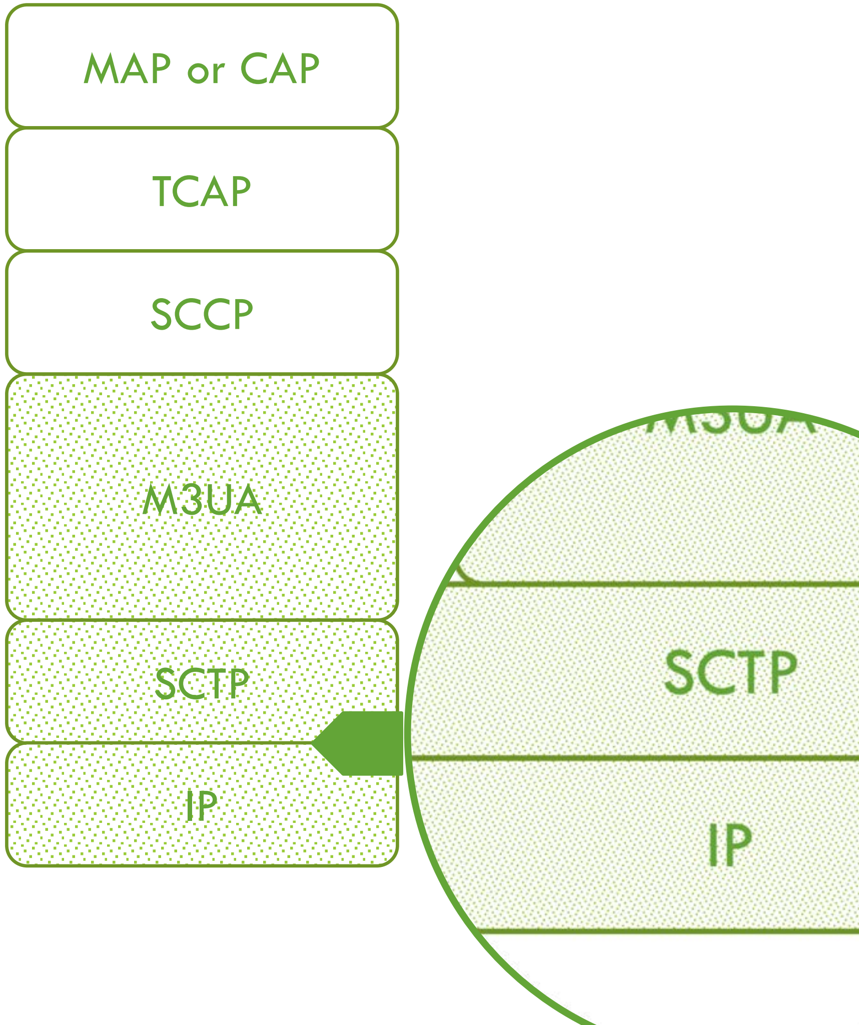
SS7 ON TOP OF IP TRANSPORT = SIGTRAN

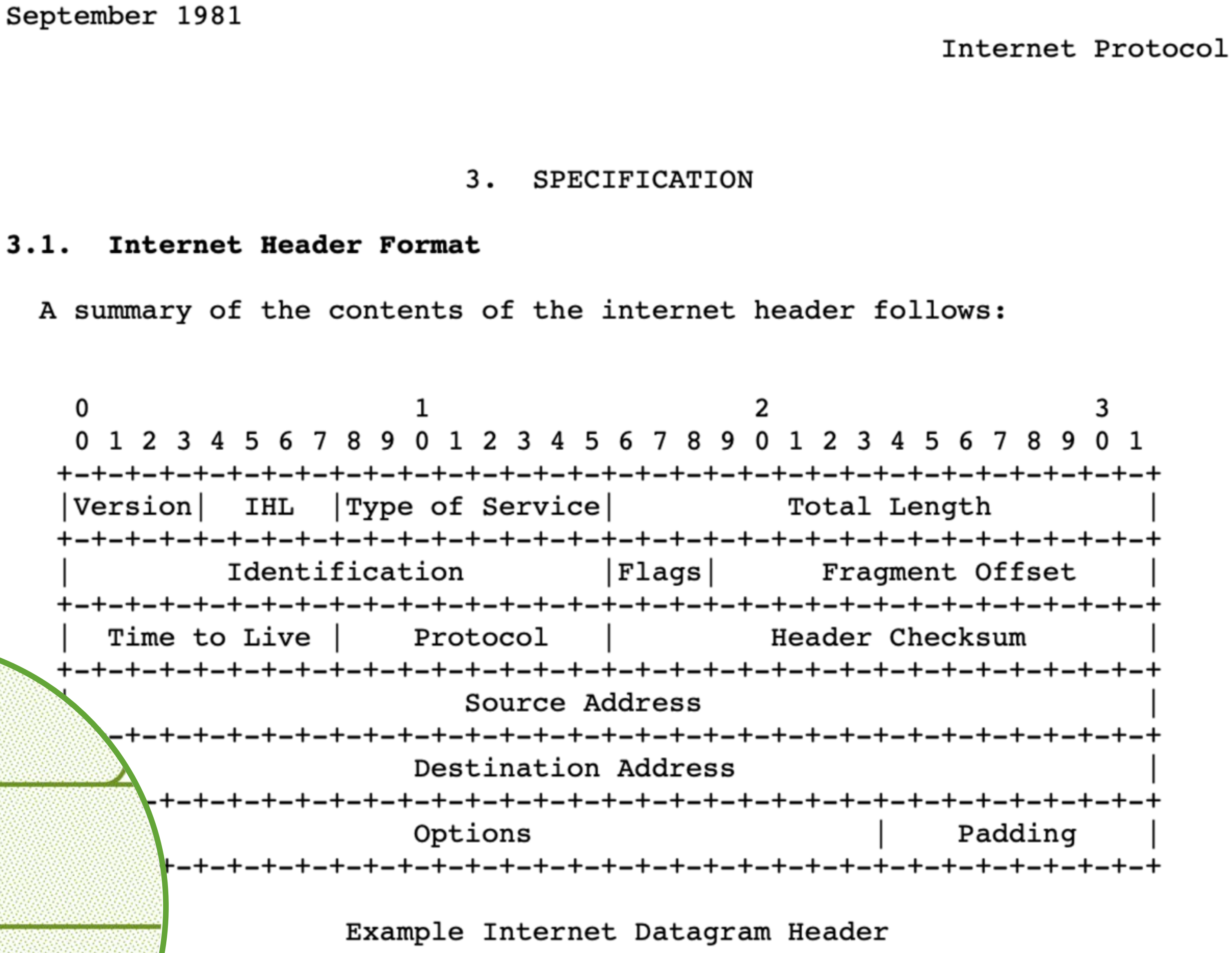


MAP or CAP
TCAP
SCCP
MTP3
MTP2
MTP1
PHY

MAP or CAP
TCAP
SCCP
M3UA
SCTP
IP
PHY

IP/SCTP




September 1981

Internet Protocol

3. SPECIFICATION

3.1. Internet Header Format

A summary of the contents of the internet header follows:

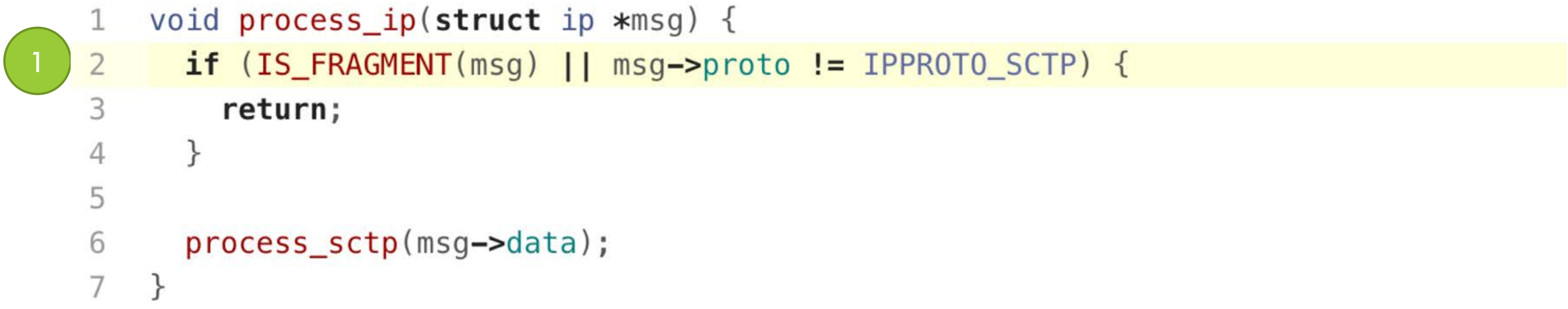
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Example Internet Datagram Header

(SOME) IP/SCTP TASKS FOR AN SS7 FIREWALL

- Check source and destination addresses are allowed to communicate
- Reassemble IP fragments, to yield to SCTP
- Reassemble SCTP fragments, to yield to M3UA

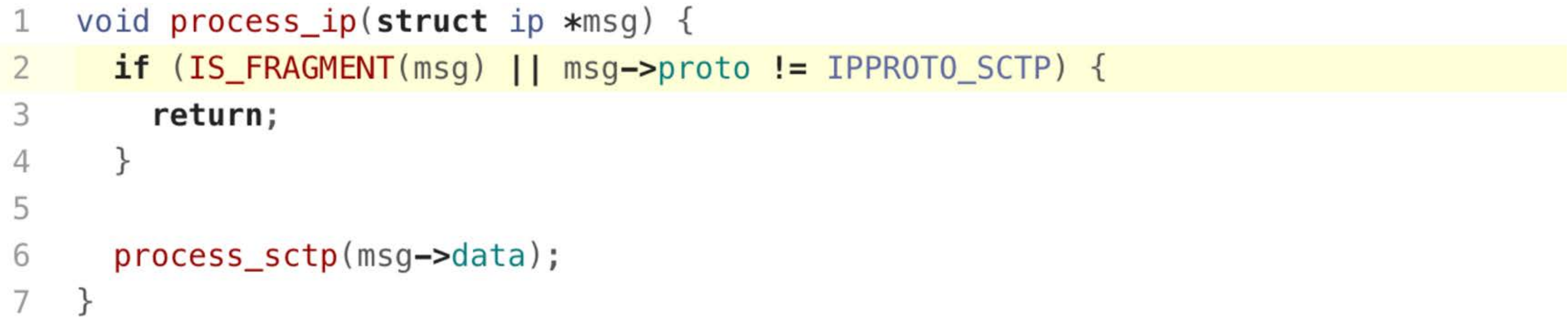
TRICK REASSEMBLY TO BYPASS DETECTION



```c
void process_ip(struct ip *msg) {
  if (IS_FRAGMENT(msg) || msg->proto != IPPROTO_SCTP) {
    return;
  }

  process_sctp(msg->data);
}
```

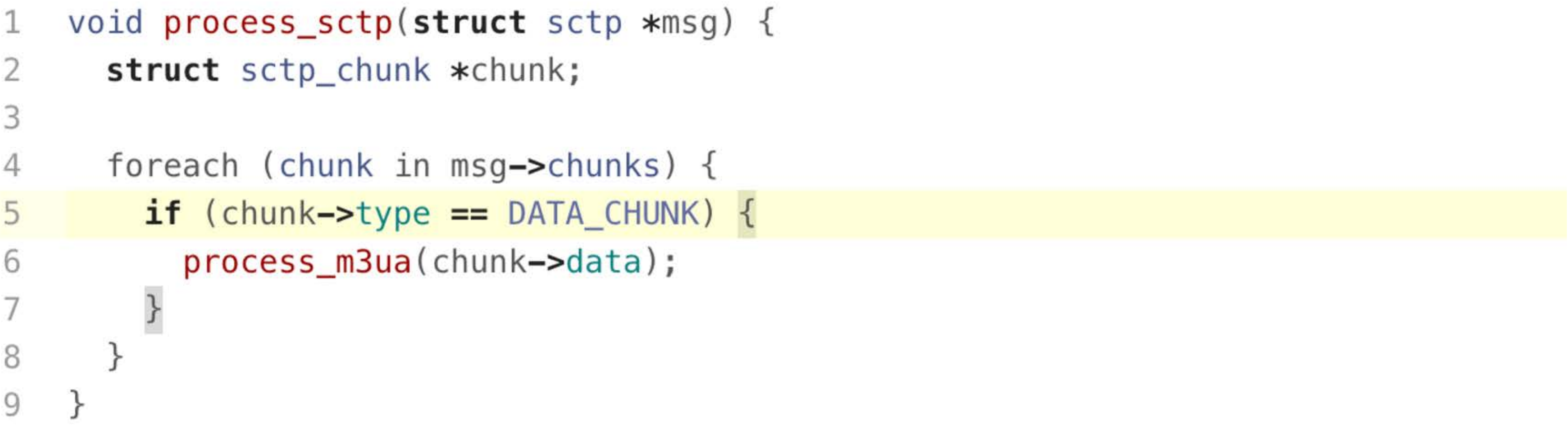
TRICK REASSEMBLY
TO BYPASS DETECTION

1



```
void process_ip(struct ip *msg) {
  if (IS_FRAGMENT(msg) || msg->proto != IPPROTO_SCTP) {
    return;
  }

  process_sctp(msg->data);
}
```

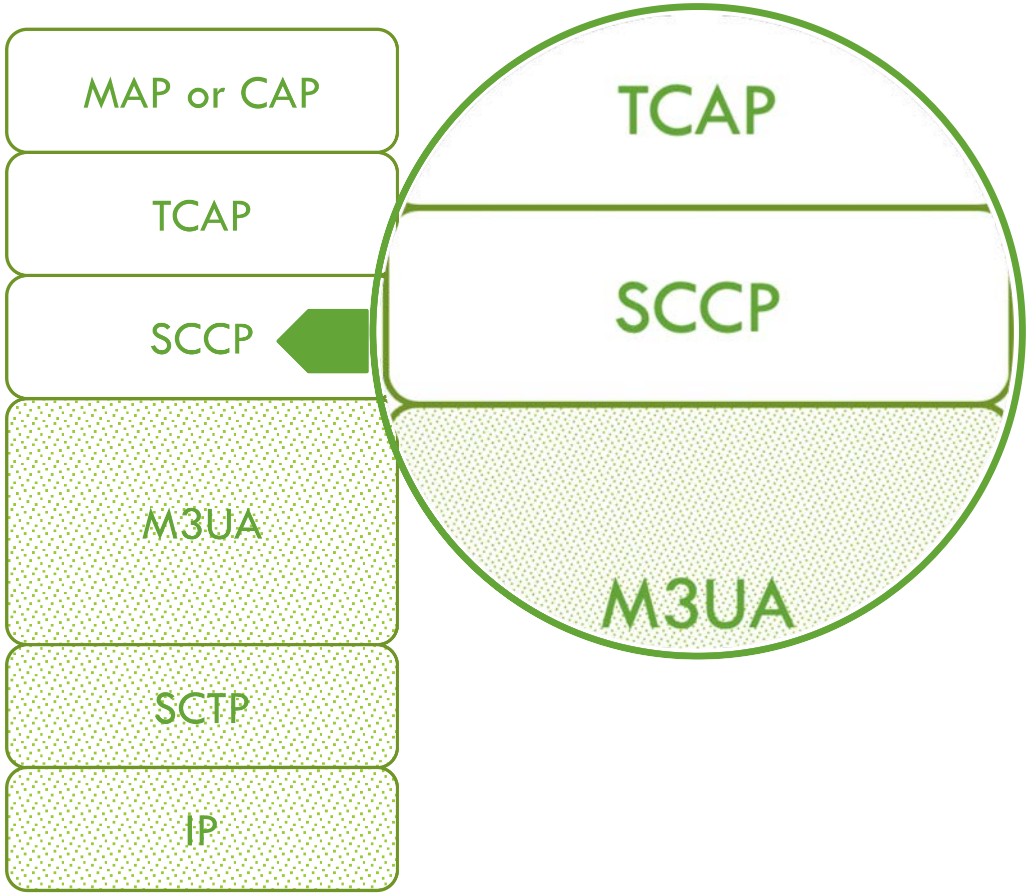
2



```
void process_sctp(struct sctp *msg) {
  struct sctp_chunk *chunk;

  foreach (chunk in msg->chunks) {
    if (chunk->type == DATA_CHUNK) {
      process_m3ua(chunk->data);
    }
  }
}
```

SCCP

SIGNALLING CONNECTION CONTROL PART



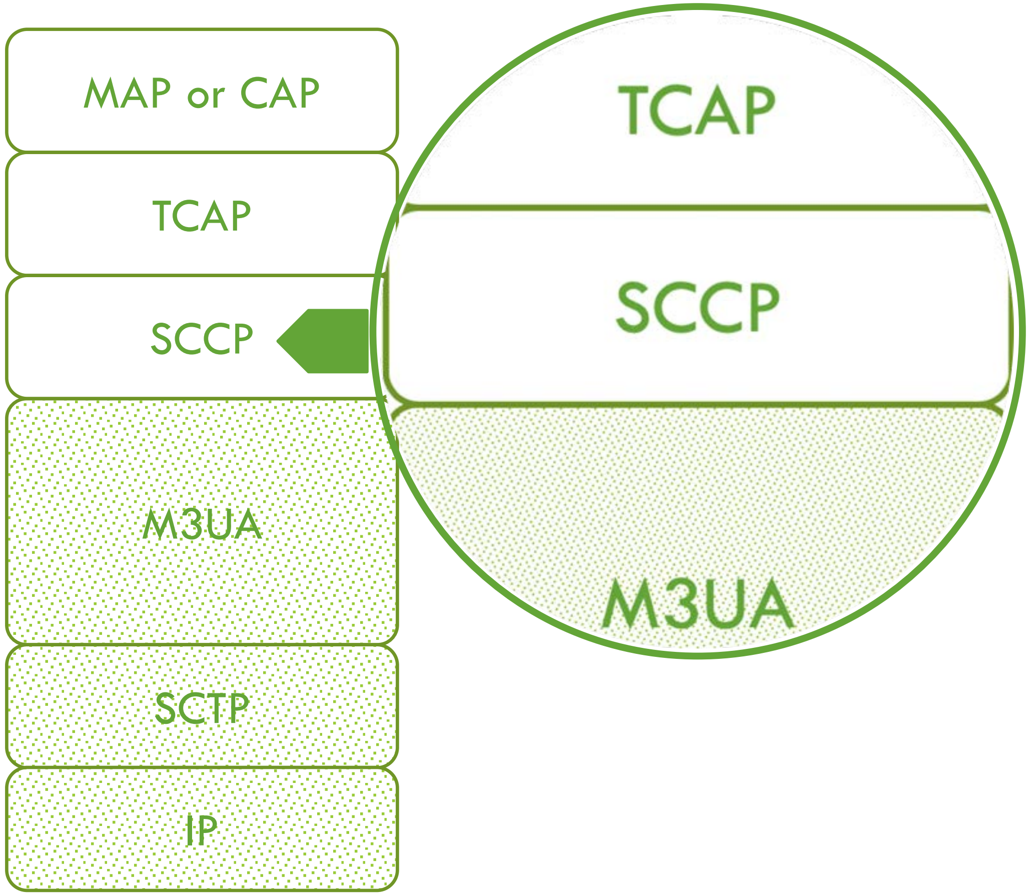
- Extends MTP routing based on point code:
  - Point code plus subsystem number
  - Or Global Title
- Provides different levels of connection
  - Management messages
  - Data messages
- Provides segmentation and reassembly

(SOME) SCCP TASKS FOR AN SS7 FIREWALL

- Retrieve called and caller addresses, to check if they are allowed to communicate
- Reassemble XUDT fragments, to further analyze a TCAP frame

SCCP

SIGNALLING CONNECTION CONTROL PART



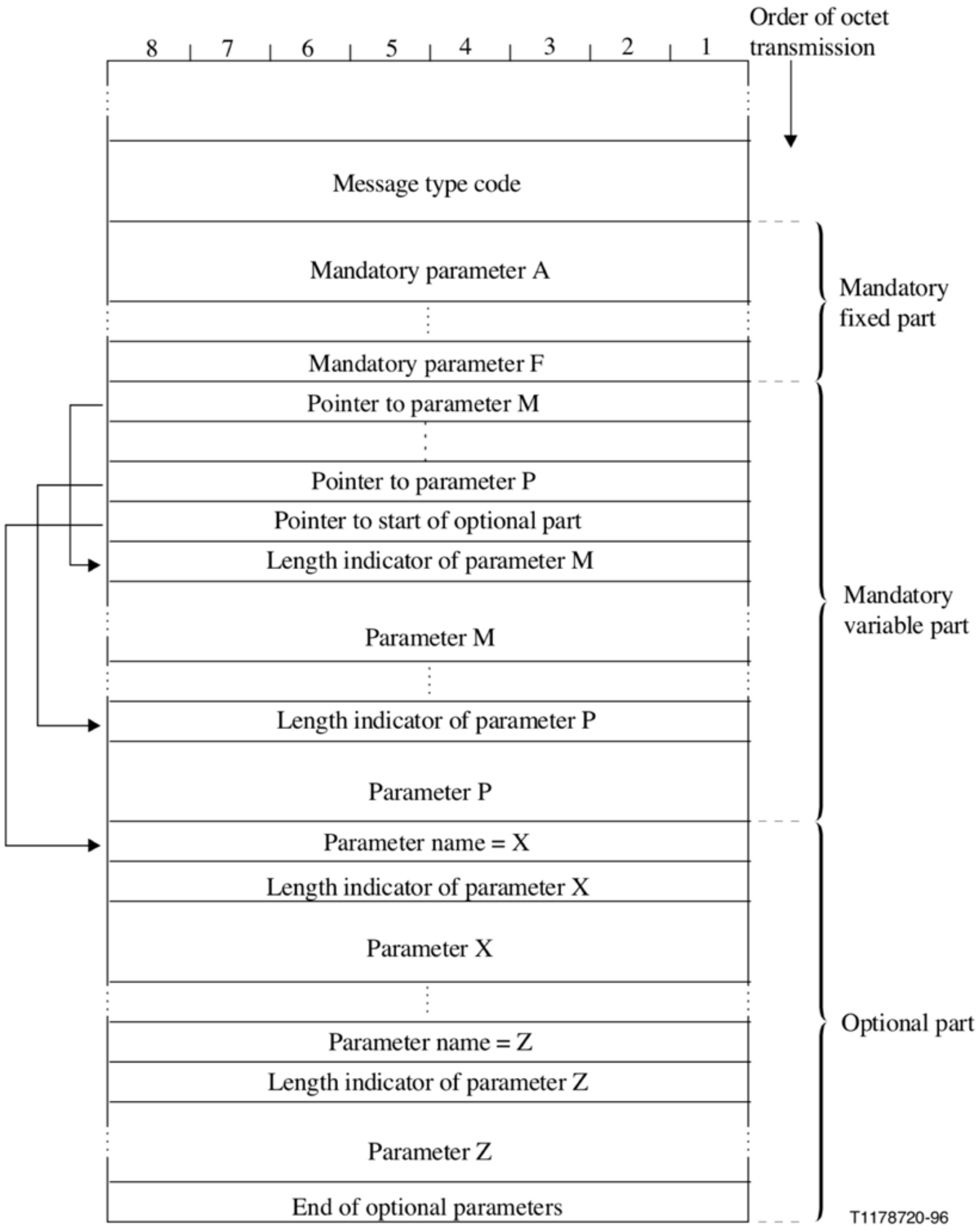
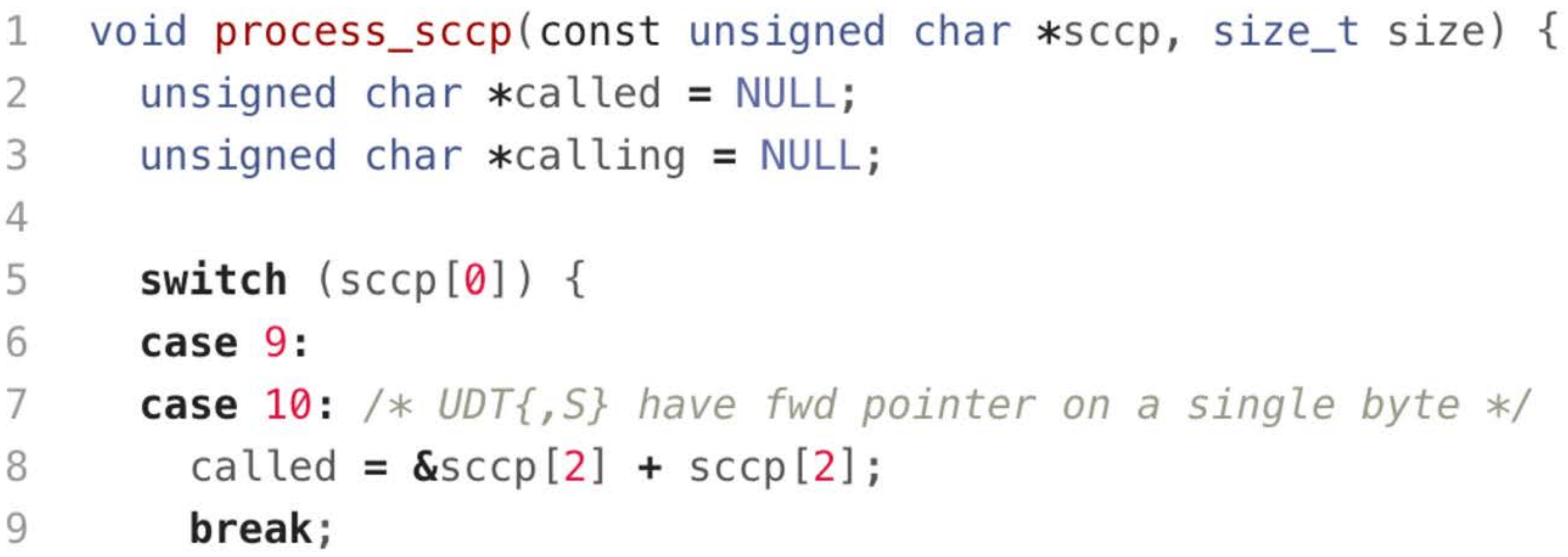


Figure 2/Q.713 – General SCCP message format

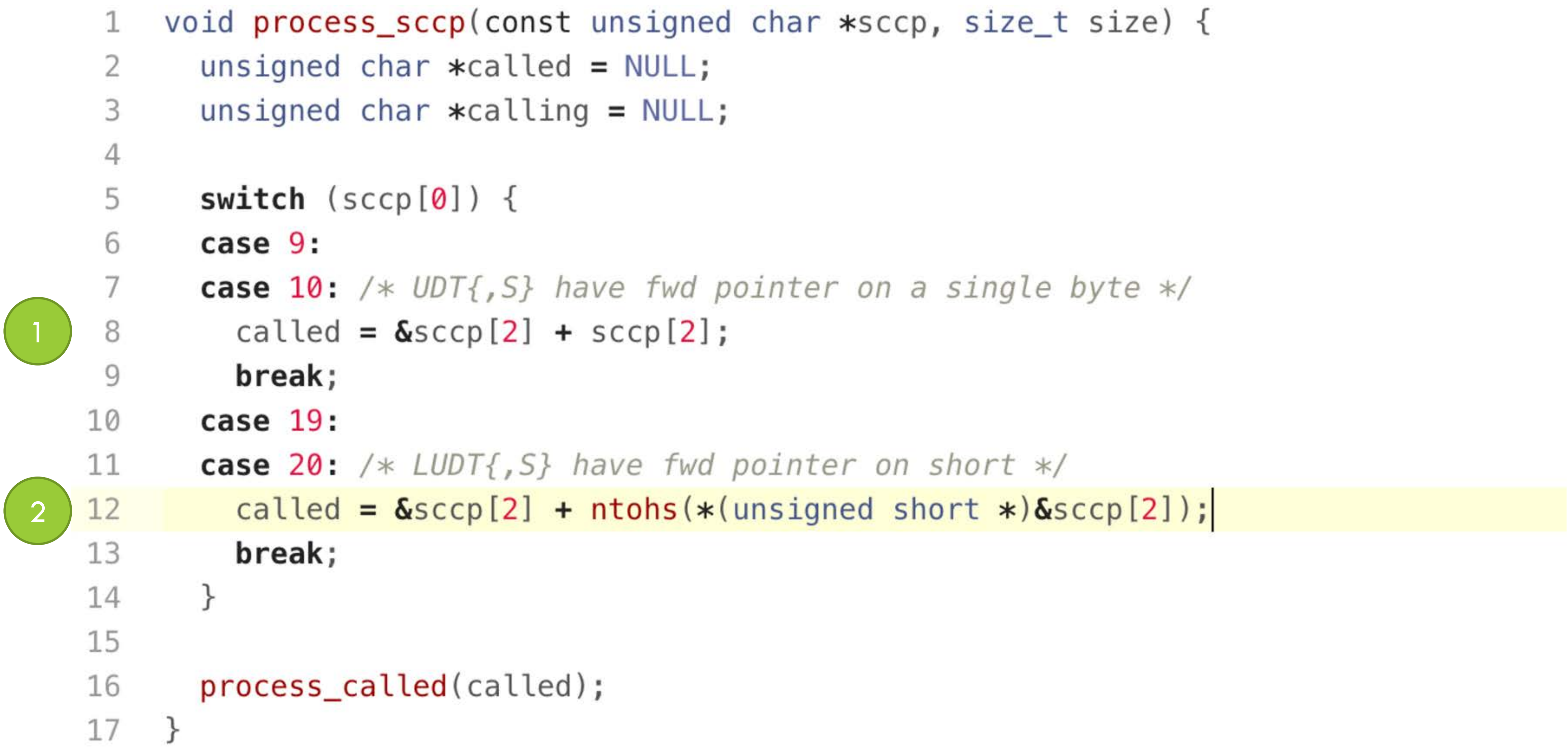
TRICK FORWARD POINTERS TO DENY SERVICE



```
void process_sccp(const unsigned char *sccp, size_t size) {
  unsigned char *called = NULL;
  unsigned char *calling = NULL;

  switch (sccp[0]) {
  case 9:
  case 10: /* UDT{,S} have fwd pointer on a single byte */
    called = &sccp[2] + sccp[2];
    break;
```

1

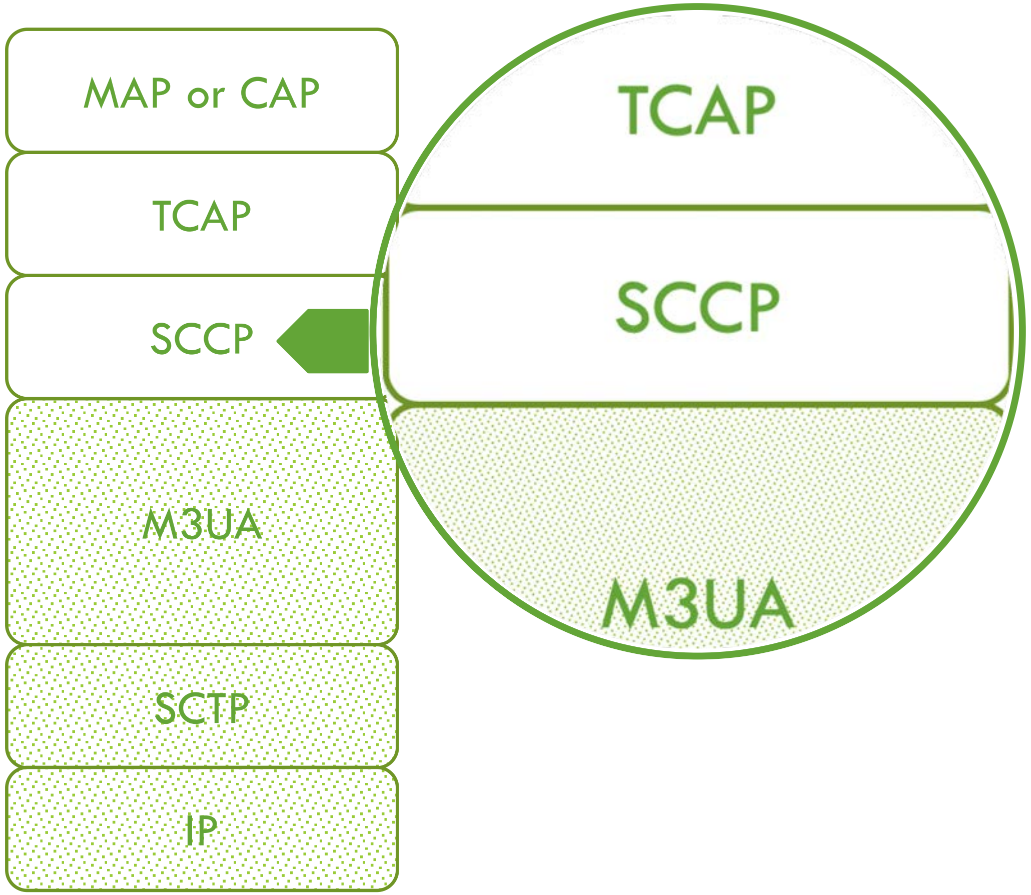
TRICK FORWARD POINTERS TO DENY SERVICE

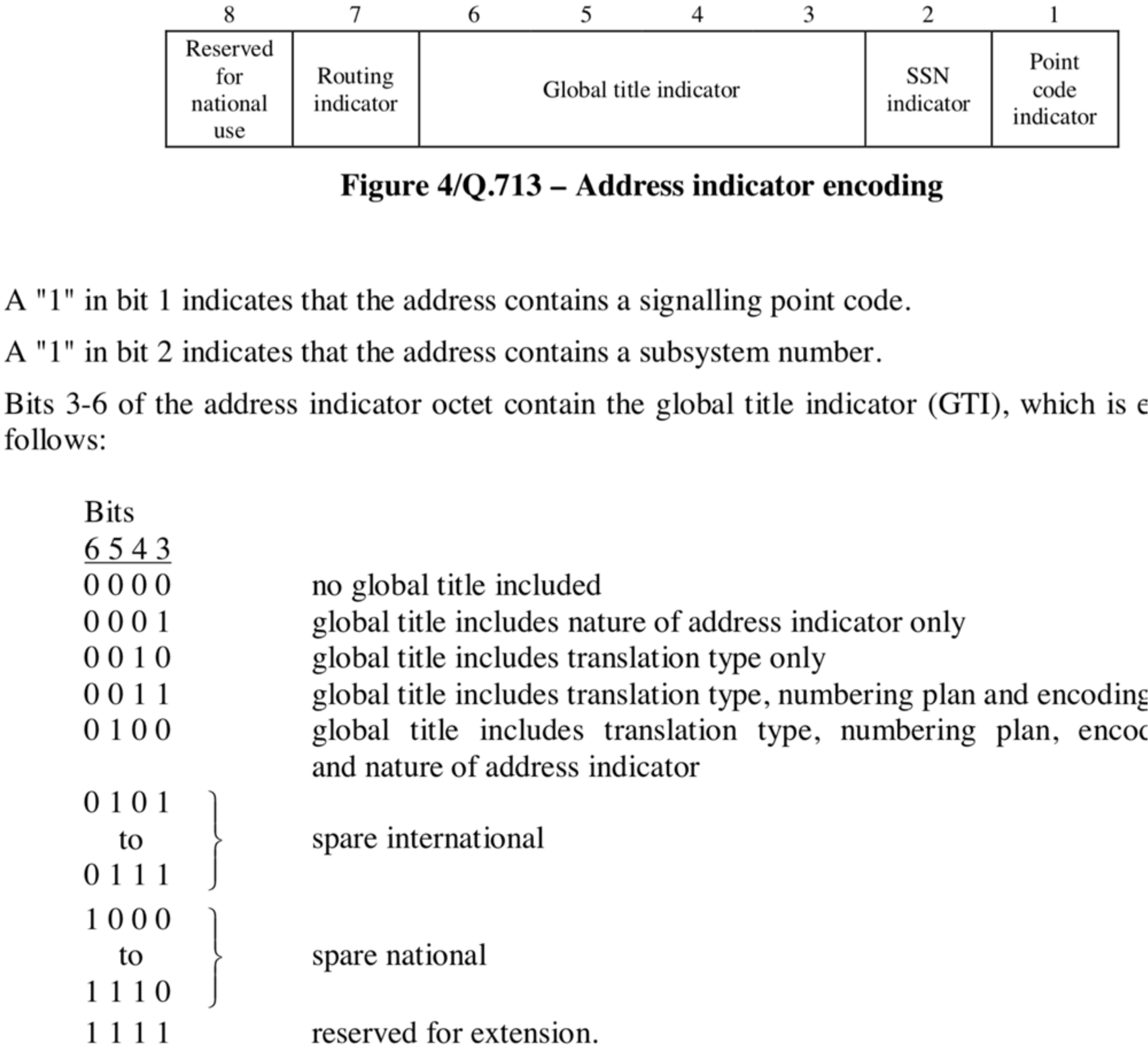


```c
void process_sccp(const unsigned char *sccp, size_t size) {
  unsigned char *called = NULL;
  unsigned char *calling = NULL;

  switch (sccp[0]) {
  case 9:
  case 10: /* UDT{,S} have fwd pointer on a single byte */
    called = &sccp[2] + sccp[2];
    break;
  case 19:
  case 20: /* LUDT{,S} have fwd pointer on short */
    called = &sccp[2] + ntohs(*(unsigned short *)&sccp[2]);
    break;
  }

  process_called(called);
}
```

SCCP

SIGNALLING CONNECTION CONTROL PART





8	7	6	5	4	3	2	1
Reserved for national use	Routing indicator	Global title indicator				SSN indicator	Point code indicator

Figure 4/Q.713 – Address indicator encoding

A "1" in bit 1 indicates that the address contains a signalling point code.

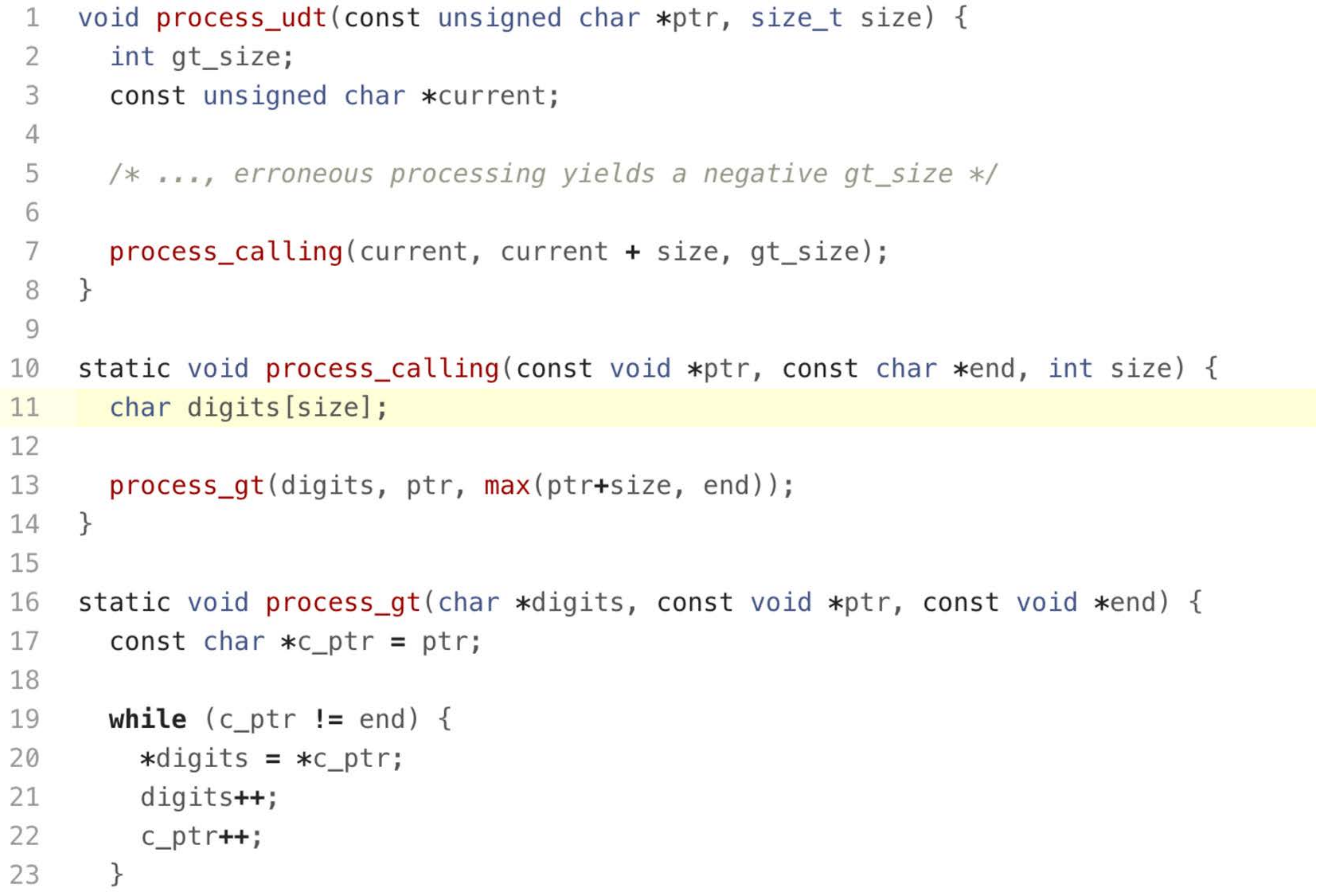
A "1" in bit 2 indicates that the address contains a subsystem number.

Bits 3-6 of the address indicator octet contain the global title indicator (GTI), which is e follows:

Bits 6 5 4 3	
0 0 0 0	no global title included
0 0 0 1	global title includes nature of address indicator only
0 0 1 0	global title includes translation type only
0 0 1 1	global title includes translation type, numbering plan and encoding
0 1 0 0	global title includes translation type, numbering plan, encod and nature of address indicator
0 1 0 1 to 0 1 1 1	spare international
1 0 0 0 to 1 1 1 0	spare national
1 1 1 1	reserved for extension.

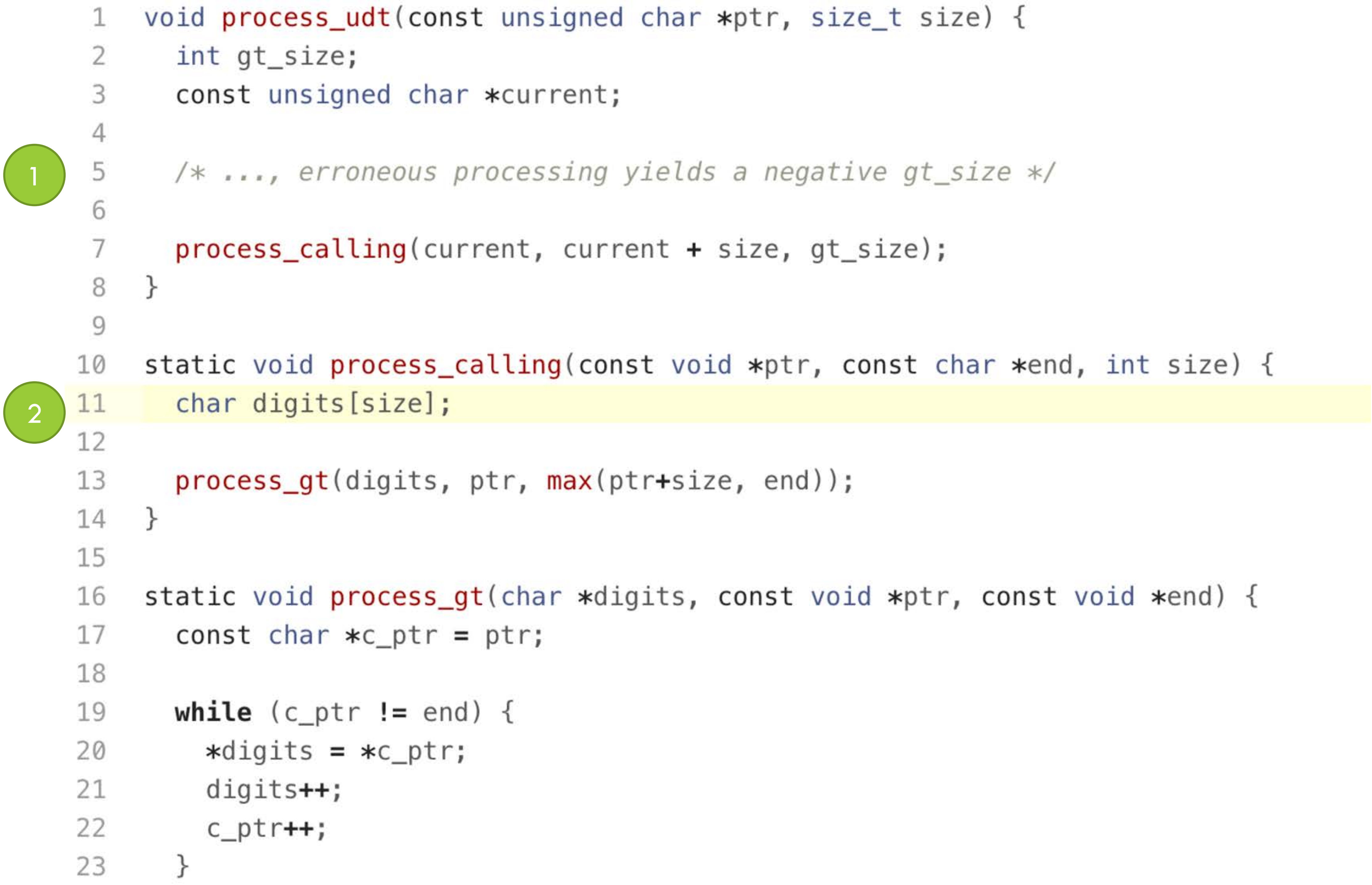
FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION

1



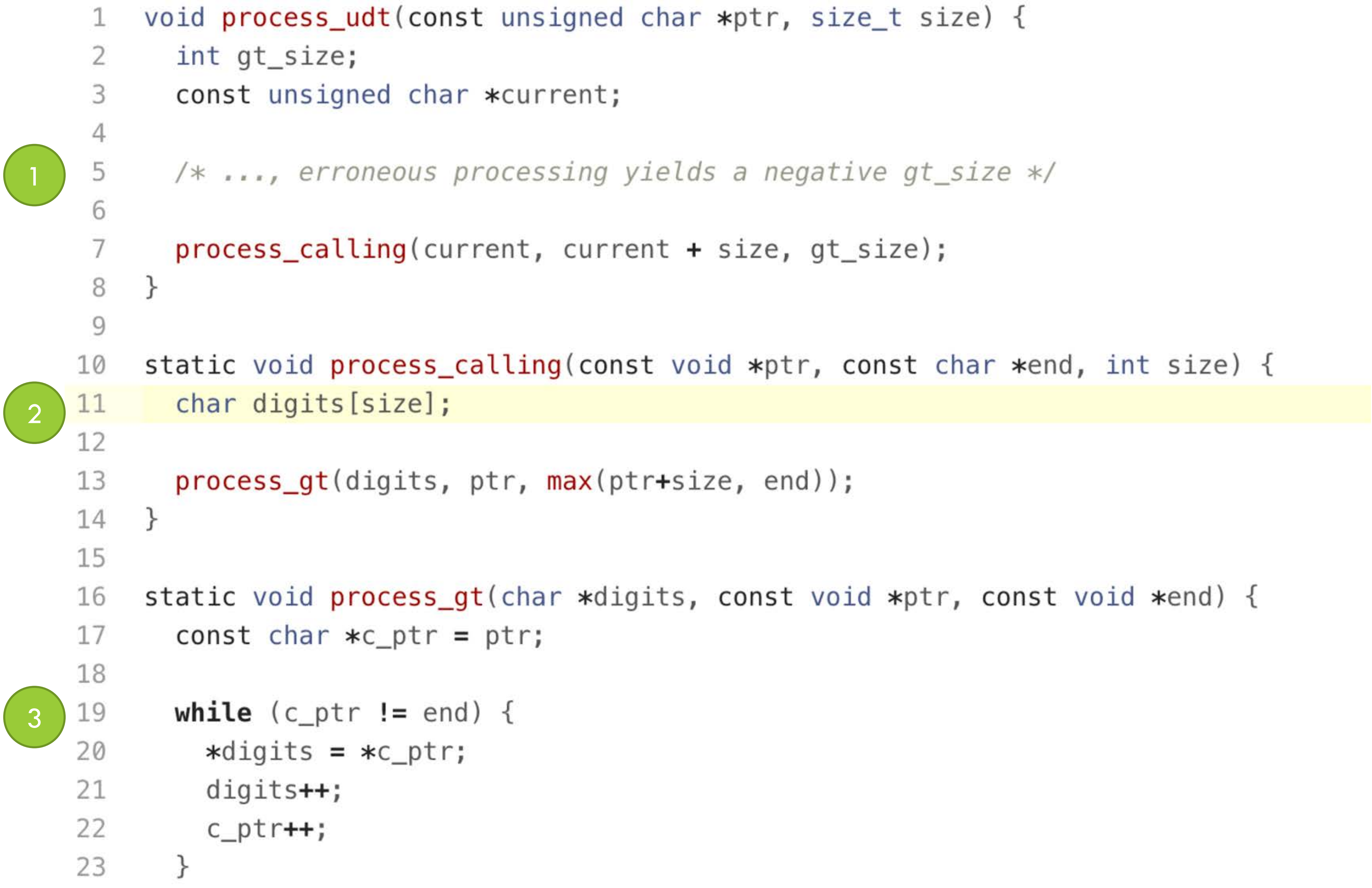
```c
void process_udt(const unsigned char *ptr, size_t size) {
  int gt_size;
  const unsigned char *current;

  /* ..., erroneous processing yields a negative gt_size */

  process_calling(current, current + size, gt_size);
}

static void process_calling(const void *ptr, const char *end, int size) {
  char digits[size];

  process_gt(digits, ptr, max(ptr+size, end));
}

static void process_gt(char *digits, const void *ptr, const void *end) {
  const char *c_ptr = ptr;

  while (c_ptr != end) {
    *digits = *c_ptr;
    digits++;
    c_ptr++;
  }
```

FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION



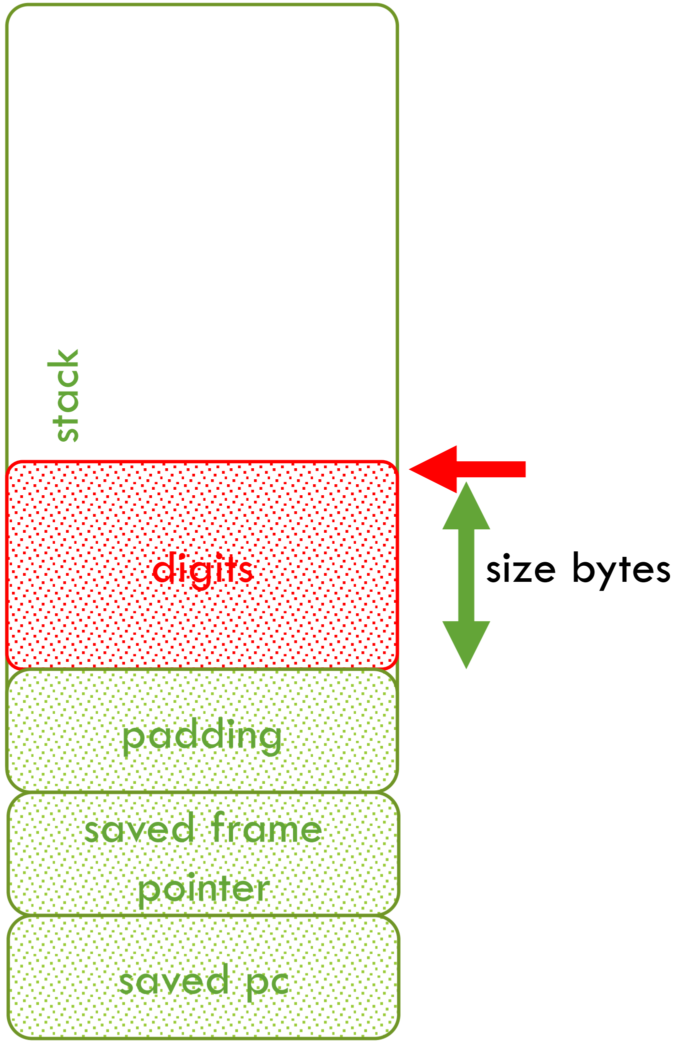
```
void process_udt(const unsigned char *ptr, size_t size) {
  int gt_size;
  const unsigned char *current;

  /* ..., erroneous processing yields a negative gt_size */

  process_calling(current, current + size, gt_size);
}

static void process_calling(const void *ptr, const char *end, int size) {
  char digits[size];

  process_gt(digits, ptr, max(ptr+size, end));
}

static void process_gt(char *digits, const void *ptr, const void *end) {
  const char *c_ptr = ptr;

  while (c_ptr != end) {
    *digits = *c_ptr;
    digits++;
    c_ptr++;
  }
```

FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION



```c
void process_udt(const unsigned char *ptr, size_t size) {
  int gt_size;
  const unsigned char *current;

  /* ..., erroneous processing yields a negative gt_size */

  process_calling(current, current + size, gt_size);
}

static void process_calling(const void *ptr, const char *end, int size) {
  char digits[size];

  process_gt(digits, ptr, max(ptr+size, end));
}

static void process_gt(char *digits, const void *ptr, const void *end) {
  const char *c_ptr = ptr;

  while (c_ptr != end) {
    *digits = *c_ptr;
    digits++;
    c_ptr++;
  }
```

FOOLING GLOBAL TITLES
TO GAIN CODE EXECUTION



stack

digits

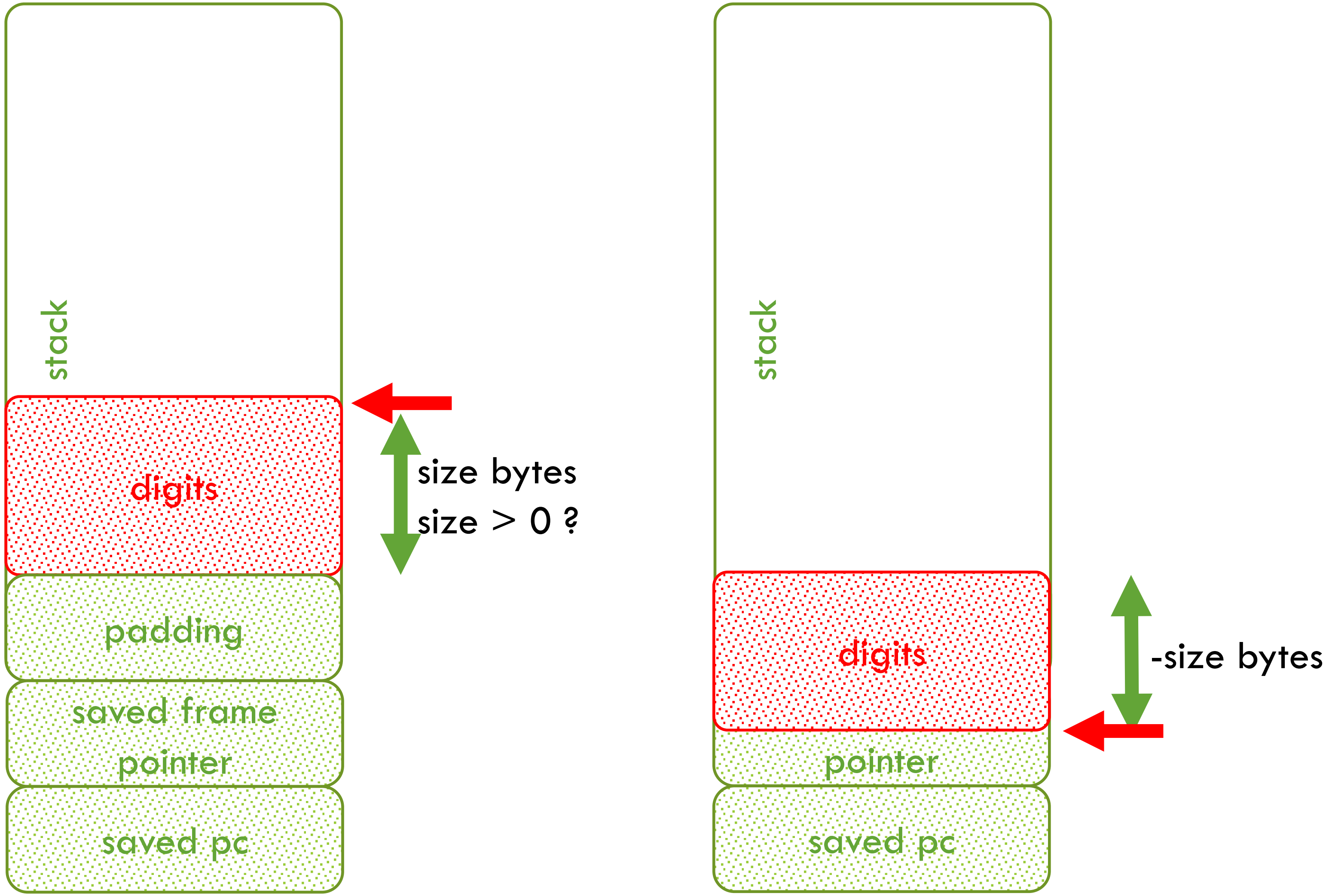
size bytes

padding

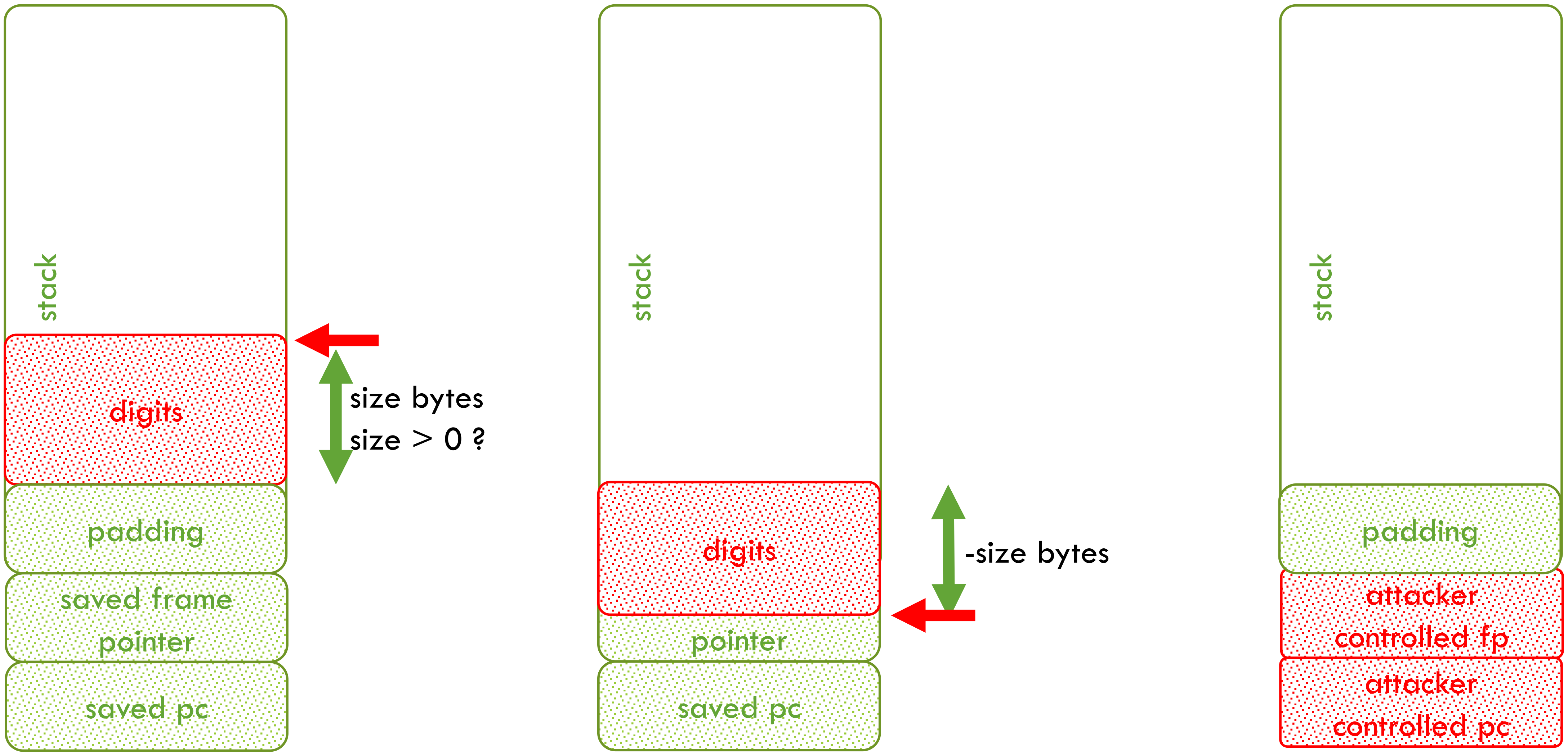
saved frame pointer

saved pc

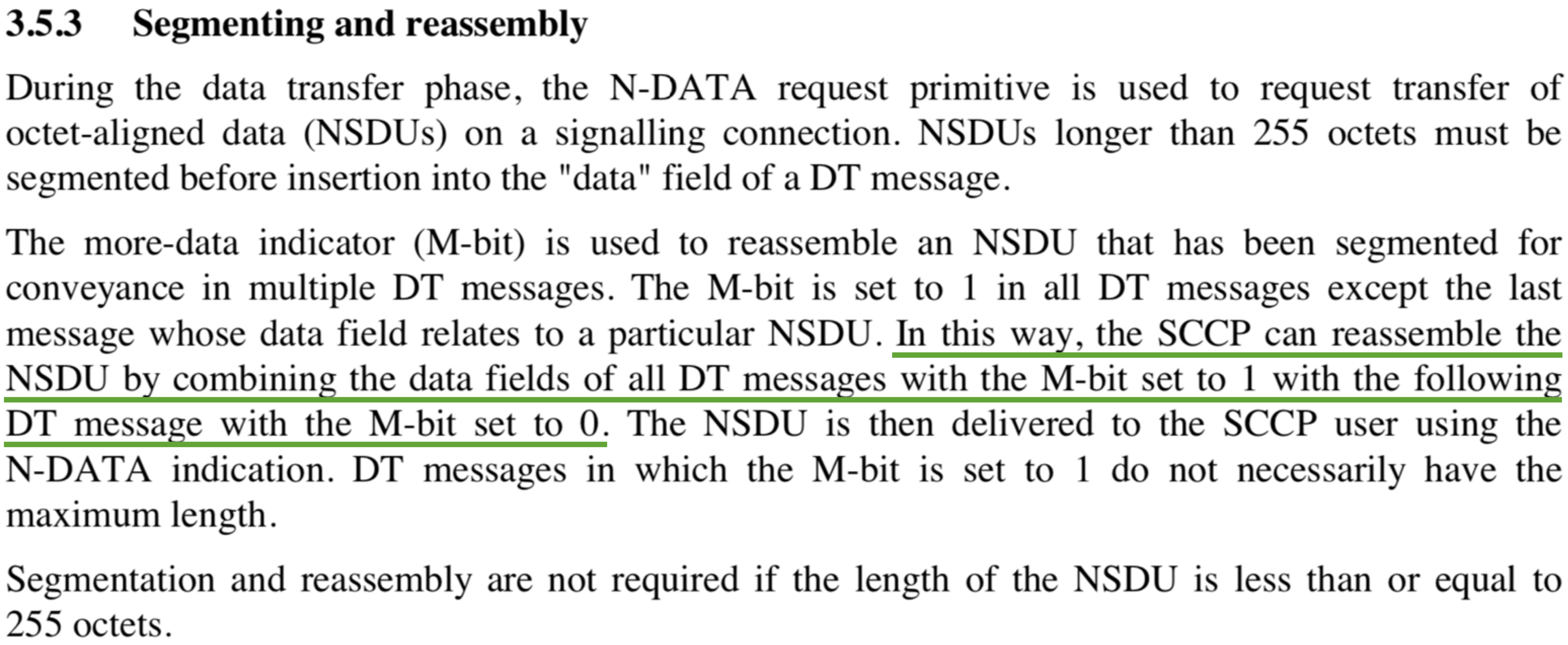
FOOLING GLOBAL TITLES
TO GAIN CODE EXECUTION



FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION



SCCP SEGMENTING & REASSEMBLY



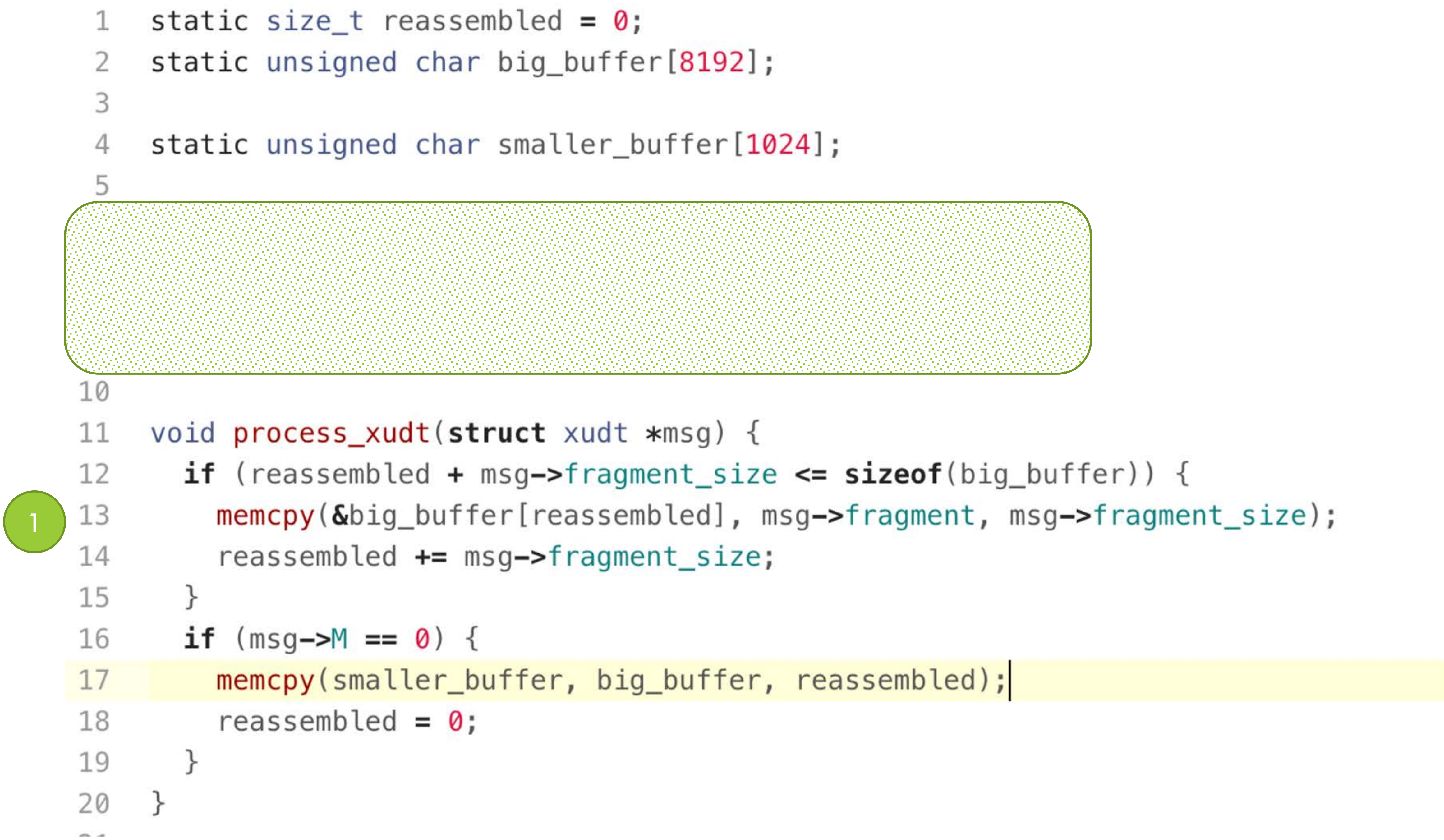
3.5.3 Segmenting and reassembly

During the data transfer phase, the N-DATA request primitive is used to request transfer of octet-aligned data (NSDUs) on a signalling connection. NSDUs longer than 255 octets must be segmented before insertion into the "data" field of a DT message.

The more-data indicator (M-bit) is used to reassemble an NSDU that has been segmented for conveyance in multiple DT messages. The M-bit is set to 1 in all DT messages except the last message whose data field relates to a particular NSDU. In this way, the SCCP can reassemble the NSDU by combining the data fields of all DT messages with the M-bit set to 1 with the following DT message with the M-bit set to 0. The NSDU is then delivered to the SCCP user using the N-DATA indication. DT messages in which the M-bit is set to 1 do not necessarily have the maximum length.

Segmentation and reassembly are not required if the length of the NSDU is less than or equal to 255 octets.

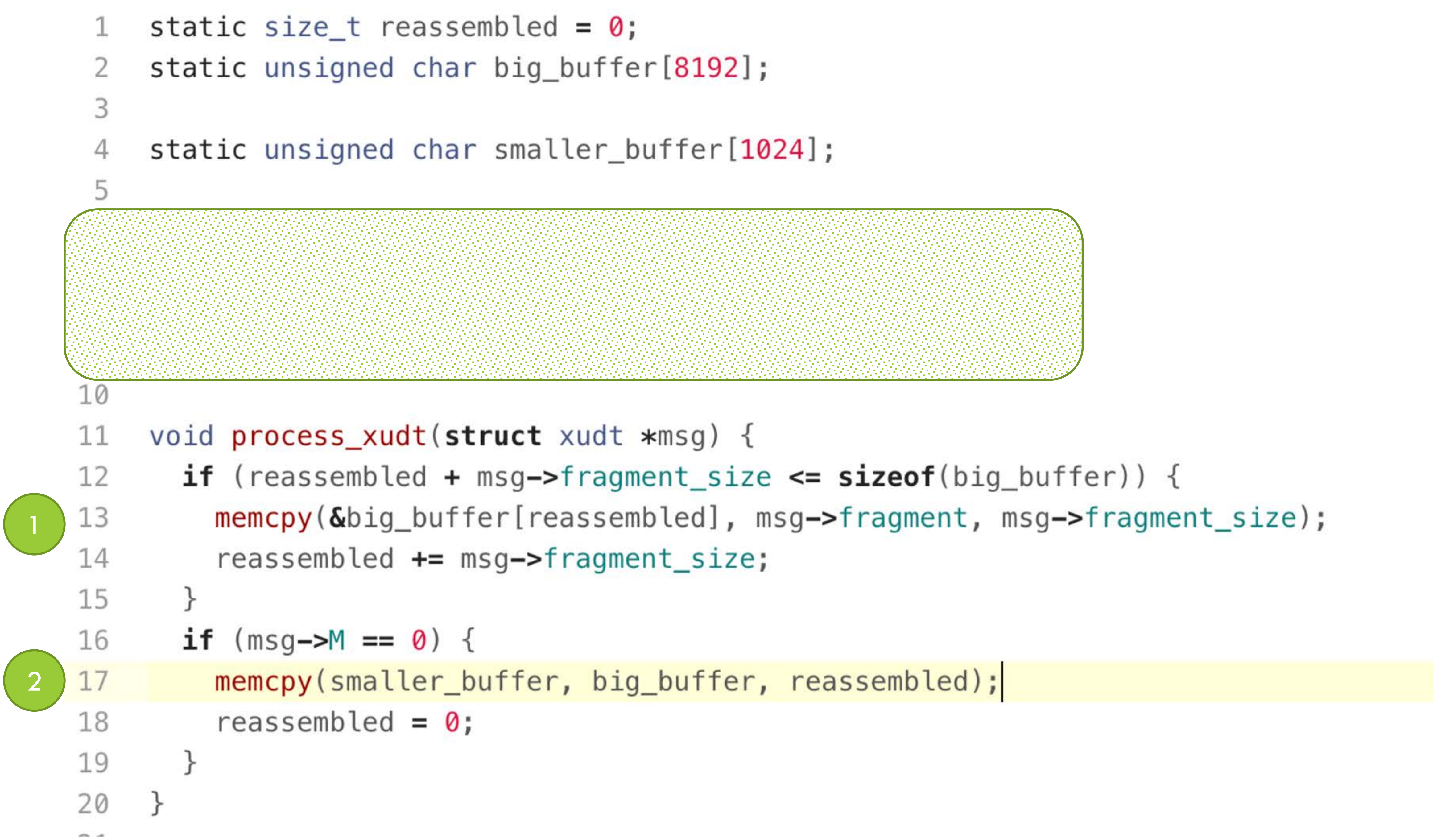
OVERFLOW REASSEMBLY
TO GAIN CODE EXECUTION



```c
static size_t reassembled = 0;
static unsigned char big_buffer[8192];

static unsigned char smaller_buffer[1024];



void process_xudt(struct xudt *msg) {
  if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
    memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
    reassembled += msg->fragment_size;
  }
  if (msg->M == 0) {
    memcpy(smaller_buffer, big_buffer, reassembled);
    reassembled = 0;
  }
}
```

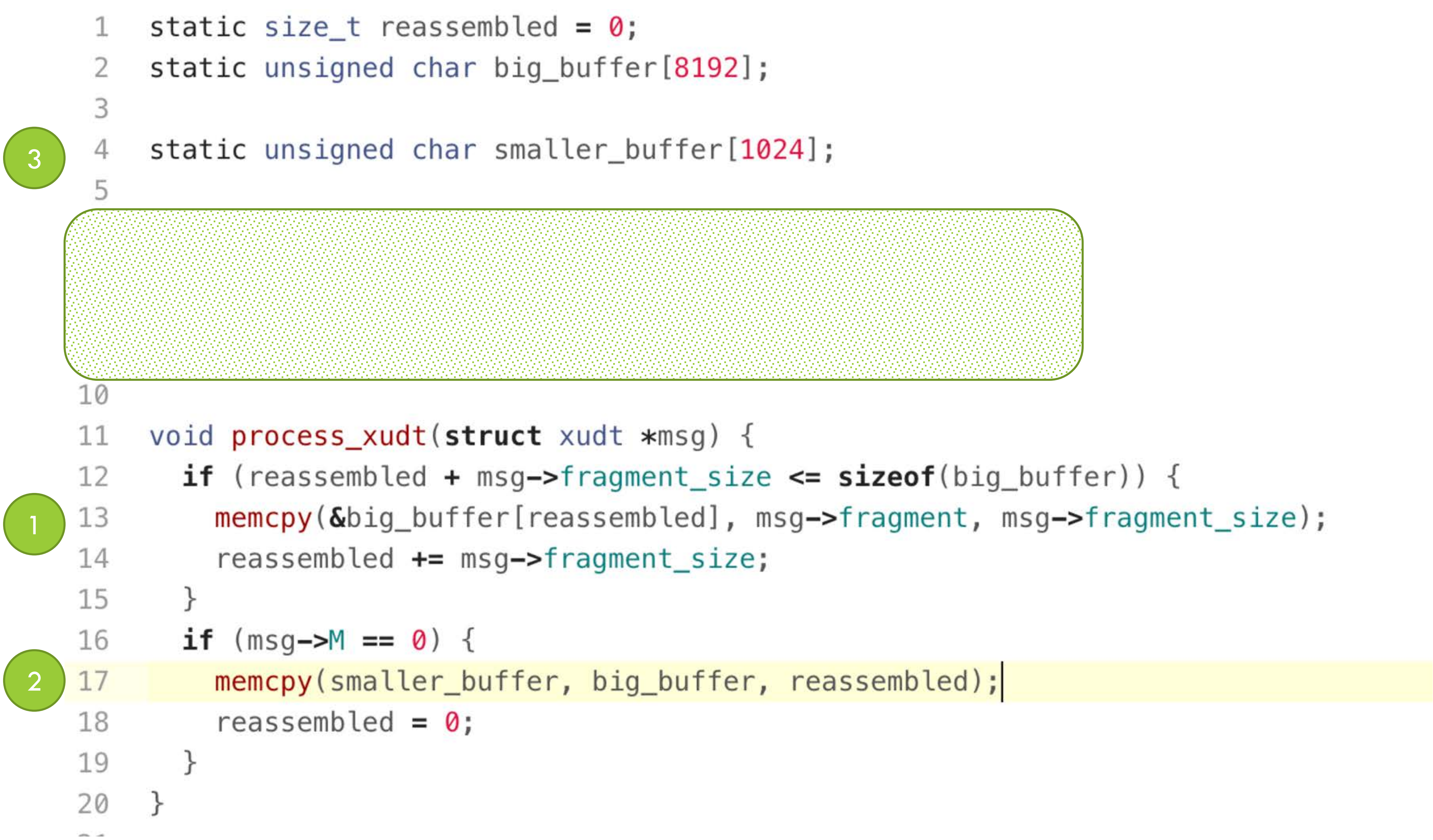
OVERFLOW REASSEMBLY

TO GAIN CODE EXECUTION



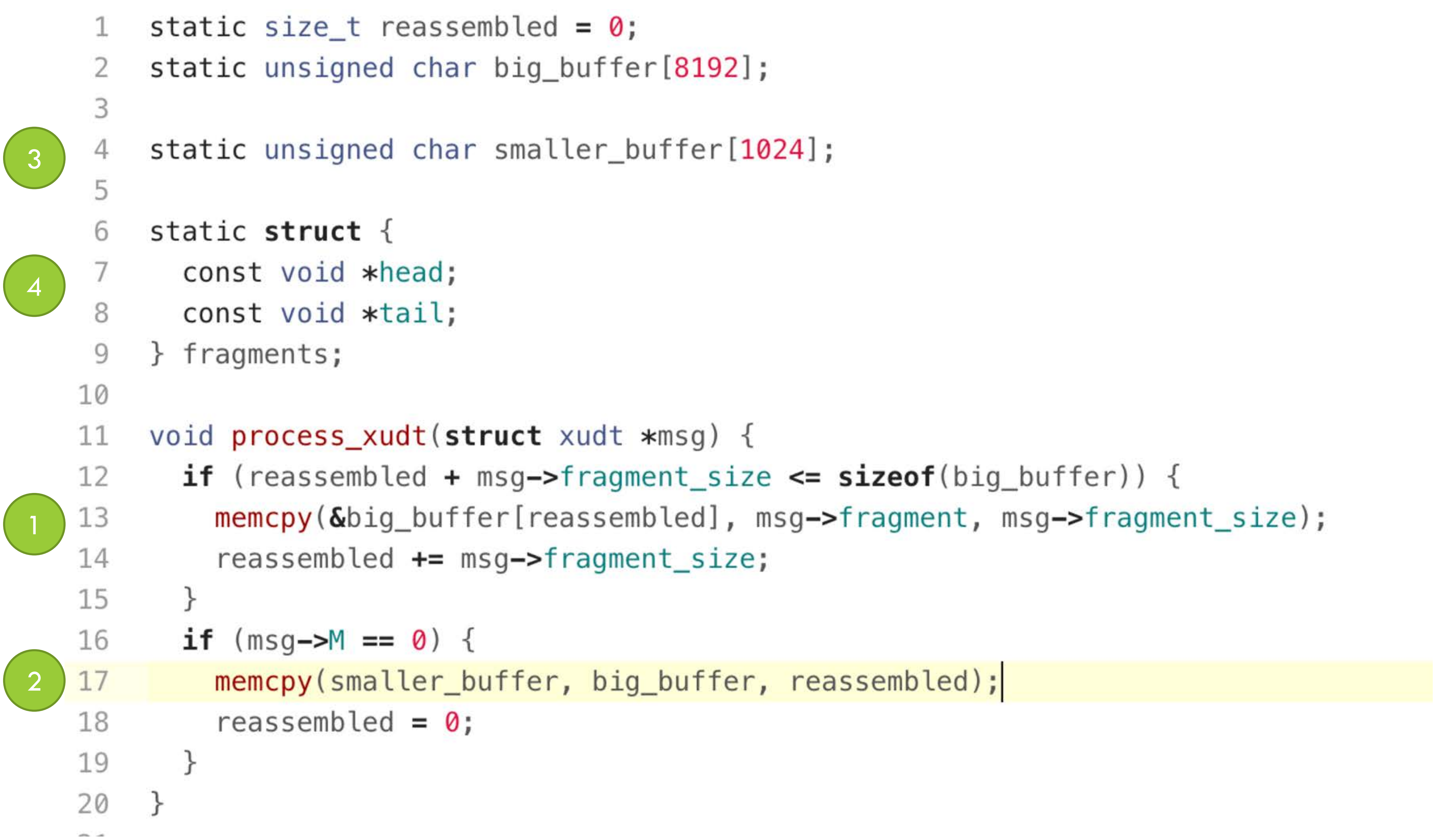
```
static size_t reassembled = 0;
static unsigned char big_buffer[8192];

static unsigned char smaller_buffer[1024];



void process_xudt(struct xudt *msg) {
  if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
    memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
    reassembled += msg->fragment_size;
  }
  if (msg->M == 0) {
    memcpy(smaller_buffer, big_buffer, reassembled);
    reassembled = 0;
  }
}
```

OVERFLOW REASSEMBLY TO GAIN CODE EXECUTION

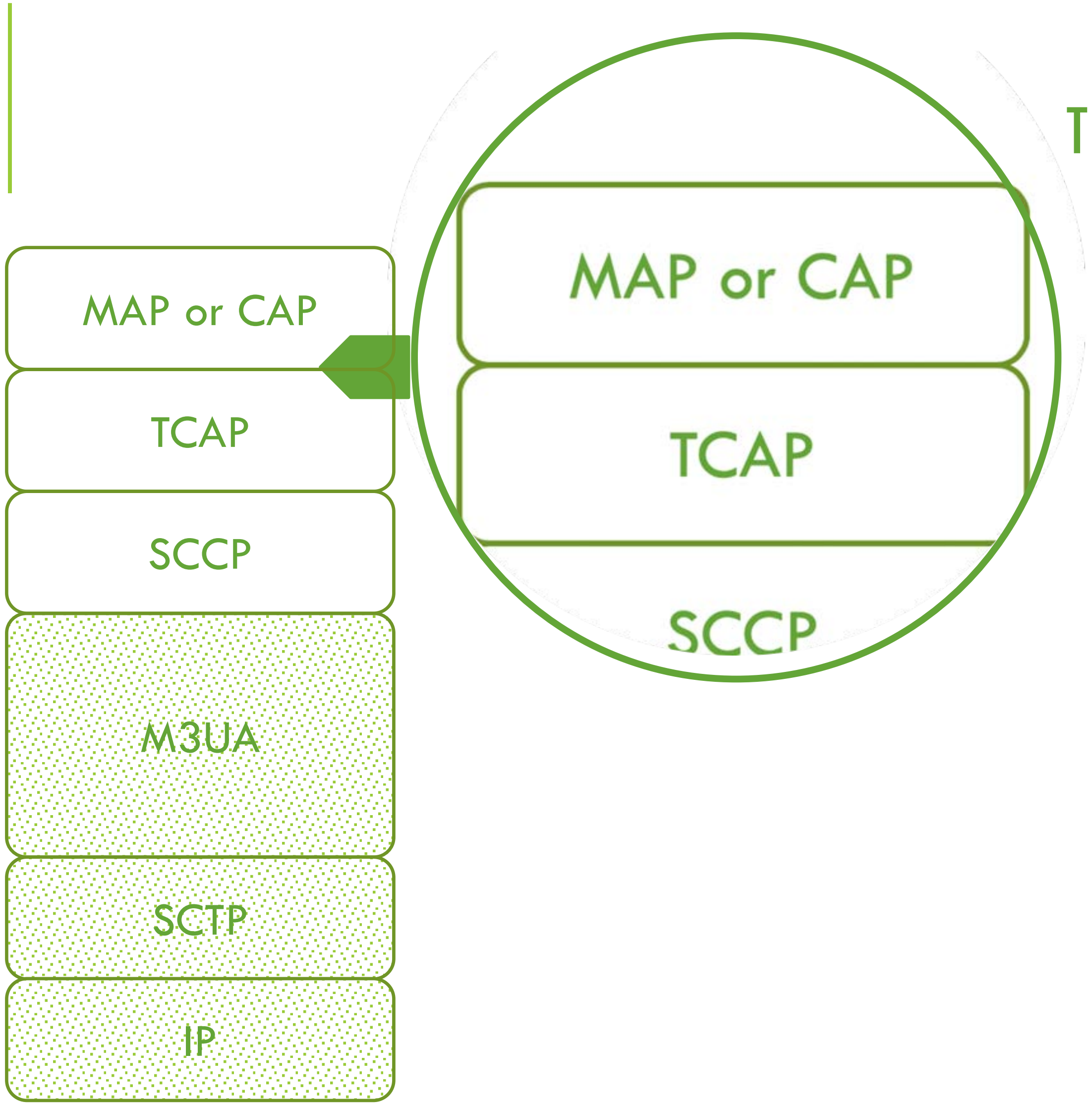


```
static size_t reassembled = 0;
static unsigned char big_buffer[8192];

static unsigned char smaller_buffer[1024];




void process_xudt(struct xudt *msg) {
  if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
    memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
    reassembled += msg->fragment_size;
  }
  if (msg->M == 0) {
    memcpy(smaller_buffer, big_buffer, reassembled);
    reassembled = 0;
  }
}
```

OVERFLOW REASSEMBLY
TO GAIN CODE EXECUTION



```
static size_t reassembled = 0;
static unsigned char big_buffer[8192];

static unsigned char smaller_buffer[1024];

static struct {
  const void *head;
  const void *tail;
} fragments;

void process_xudt(struct xudt *msg) {
  if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
    memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
    reassembled += msg->fragment_size;
  }
  if (msg->M == 0) {
    memcpy(smaller_buffer, big_buffer, reassembled);
    reassembled = 0;
  }
}
```



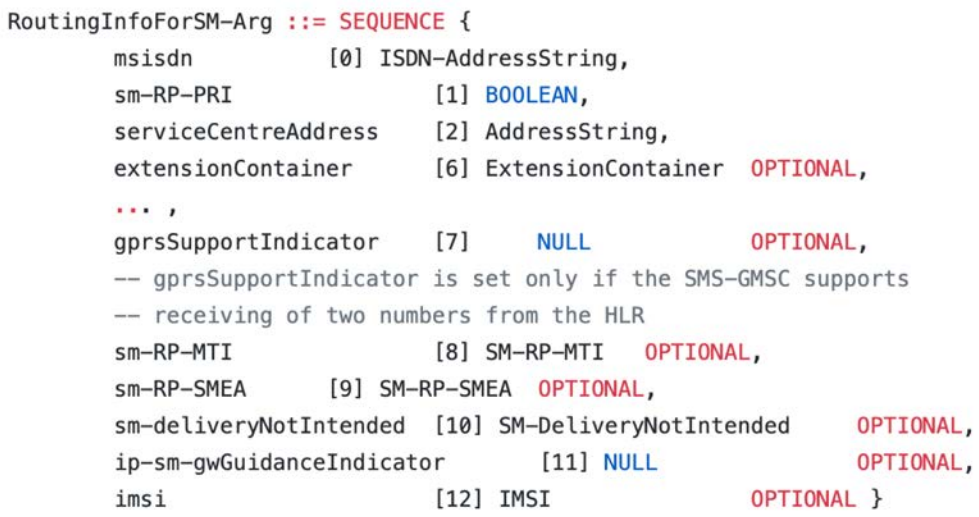
TCAP, MAP & CAP

TRANSACTION CAPABILITIES APPLICATION PART
MOBILE APPLICATION PART

- TCAP provides dialog semantics
  - With indication of upper application in an Application Context Name
- MAP provides application to mobile core nodes, using multiple operations
  - Short message service
  - Call handling
  - Mobility
  - …
- Specified in ASN.1, encoded in BER

ASN.1 SHIELDS FROM PROGRAMMING ERRORS
ABSTRACT NOTATION

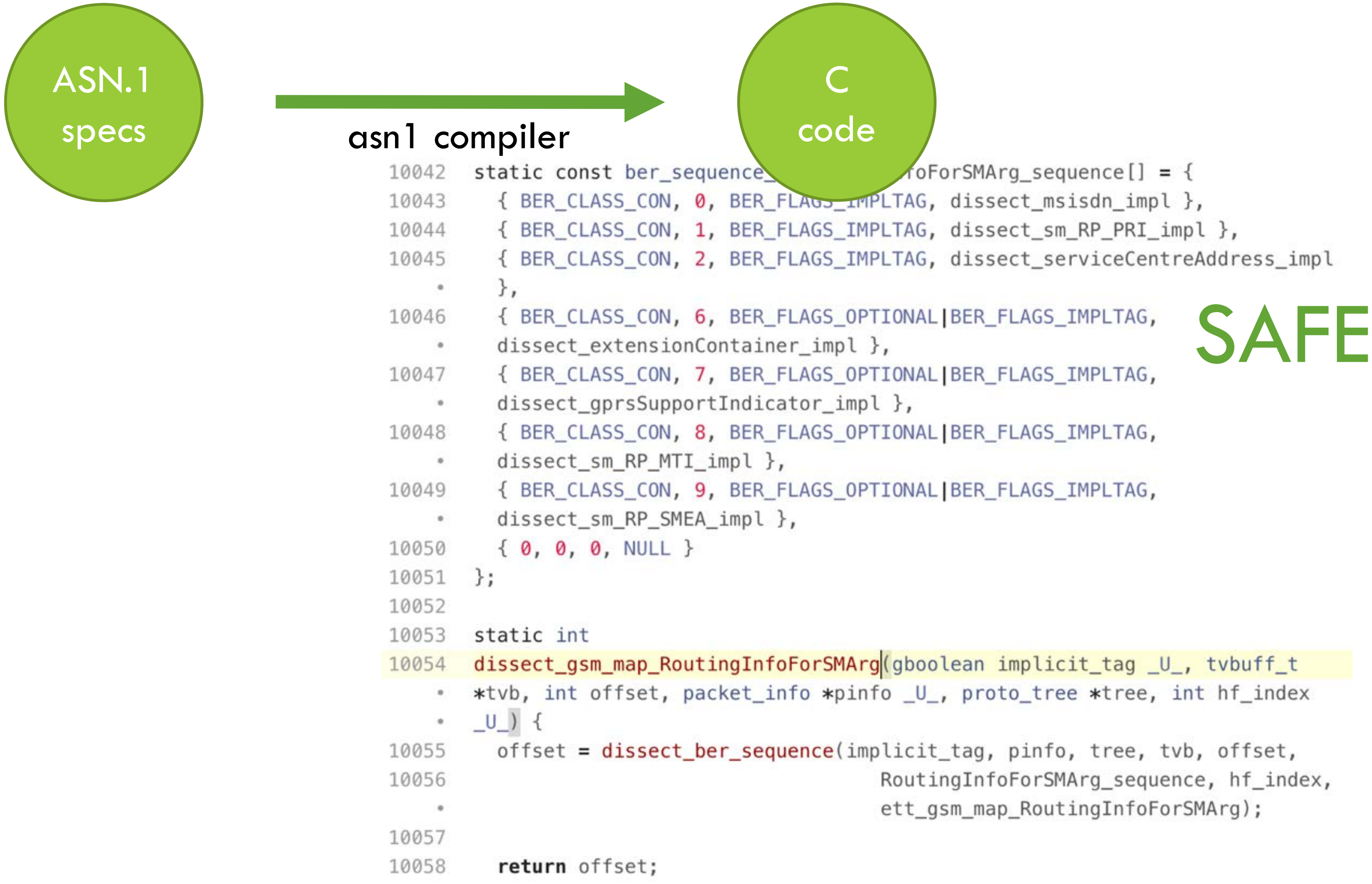
ASN.1
specs



```
RoutingInfoForSM-Arg ::= SEQUENCE {
        msisdn          [0] ISDN-AddressString,
        sm-RP-PRI                [1] BOOLEAN,
        serviceCentreAddress     [2] AddressString,
        extensionContainer       [6] ExtensionContainer  OPTIONAL,
        ... ,
        gprsSupportIndicator     [7]     NULL            OPTIONAL,
        -- gprsSupportIndicator is set only if the SMS-GMSC supports
        -- receiving of two numbers from the HLR
        sm-RP-MTI                [8] SM-RP-MTI   OPTIONAL,
        sm-RP-SMEA      [9] SM-RP-SMEA  OPTIONAL,
        sm-deliveryNotIntended  [10] SM-DeliveryNotIntended     OPTIONAL,
        ip-sm-gwGuidanceIndicator       [11] NULL               OPTIONAL,
        imsi                     [12] IMSI              OPTIONAL }
```

ASN.1 SHIELDS FROM PROGRAMMING ERRORS

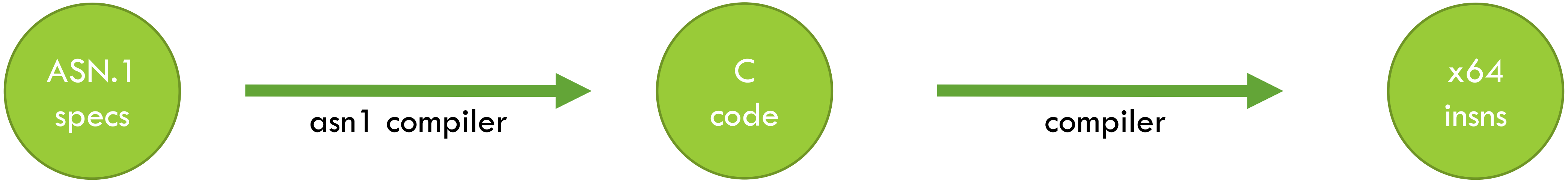
GENERATE ENCODER AND DECODER SOURCE CODE

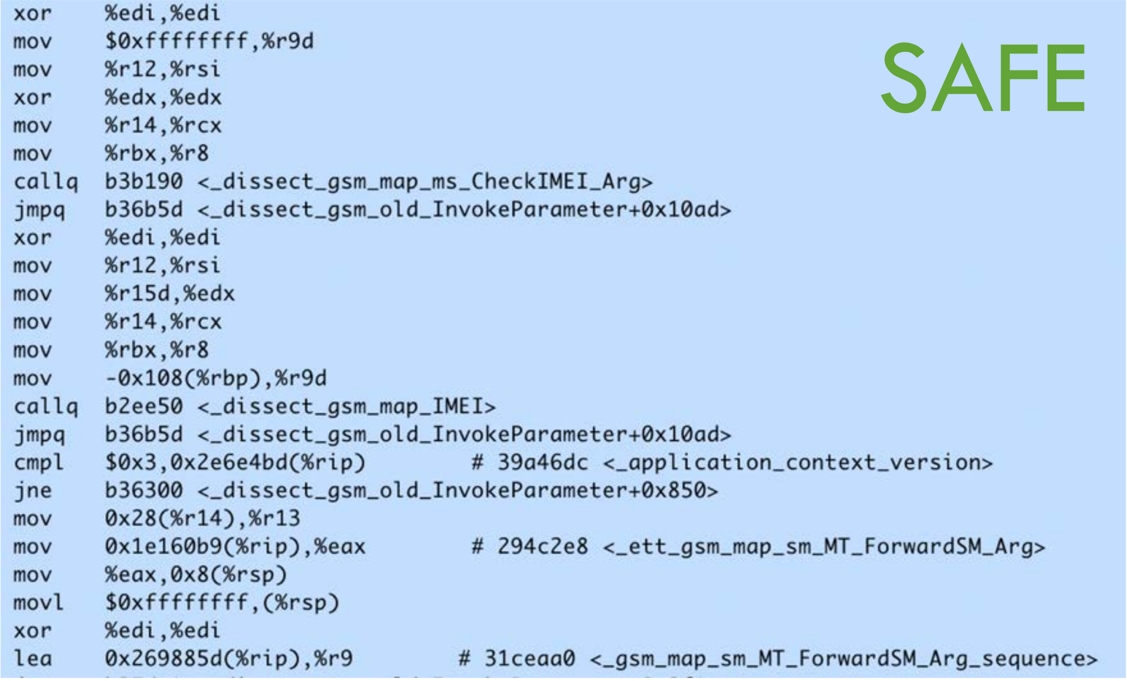


```
static const ber_sequence_[illegible]oForSMArg_sequence[] = {
  { BER_CLASS_CON, 0, BER_FLAGS_IMPLTAG, dissect_msisdn_impl },
  { BER_CLASS_CON, 1, BER_FLAGS_IMPLTAG, dissect_sm_RP_PRI_impl },
  { BER_CLASS_CON, 2, BER_FLAGS_IMPLTAG, dissect_serviceCentreAddress_impl
  },
  { BER_CLASS_CON, 6, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,
  dissect_extensionContainer_impl },
  { BER_CLASS_CON, 7, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,
  dissect_gprsSupportIndicator_impl },
  { BER_CLASS_CON, 8, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,
  dissect_sm_RP_MTI_impl },
  { BER_CLASS_CON, 9, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,
  dissect_sm_RP_SMEA_impl },
  { 0, 0, 0, NULL }
};

static int
dissect_gsm_map_RoutingInfoForSMArg(gboolean implicit_tag _U_, tvbuff_t
*tvb, int offset, packet_info *pinfo _U_, proto_tree *tree, int hf_index
_U_) {
  offset = dissect_ber_sequence(implicit_tag, pinfo, tree, tvb, offset,
                                 RoutingInfoForSMArg_sequence, hf_index,
                                 ett_gsm_map_RoutingInfoForSMArg);

  return offset;
```

SAFE

ASN.1 SHIELDS FROM PROGRAMMING ERRORS
SAFE MACHINE CODE

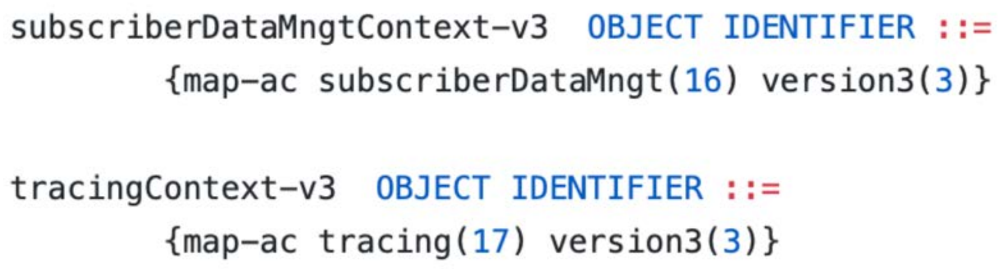



SAFE

```
xor     %edi,%edi
mov     $0xffffffff,%r9d
mov     %r12,%rsi
xor     %edx,%edx
mov     %r14,%rcx
mov     %rbx,%r8
callq   b3b190 <_dissect_gsm_map_ms_CheckIMEI_Arg>
jmpq    b36b5d <_dissect_gsm_old_InvokeParameter+0x10ad>
xor     %edi,%edi
mov     %r12,%rsi
mov     %r15d,%edx
mov     %r14,%rcx
mov     %rbx,%r8
mov     -0x108(%rbp),%r9d
callq   b2ee50 <_dissect_gsm_map_IMEI>
jmpq    b36b5d <_dissect_gsm_old_InvokeParameter+0x10ad>
cmpl    $0x3,0x2e6e4bd(%rip)        # 39a46dc <_application_context_version>
jne     b36300 <_dissect_gsm_old_InvokeParameter+0x850>
mov     0x28(%r14),%r13
mov     0x1e160b9(%rip),%eax        # 294c2e8 <_ett_gsm_map_sm_MT_ForwardSM_Arg>
mov     %eax,0x8(%rsp)
movl    $0xffffffff,(%rsp)
xor     %edi,%edi
lea     0x269885d(%rip),%r9        # 31ceaa0 <_gsm_map_sm_MT_ForwardSM_Arg_sequence>
```

(SOME) TCAP/MAP TASKS FOR AN SS7 FIREWALL

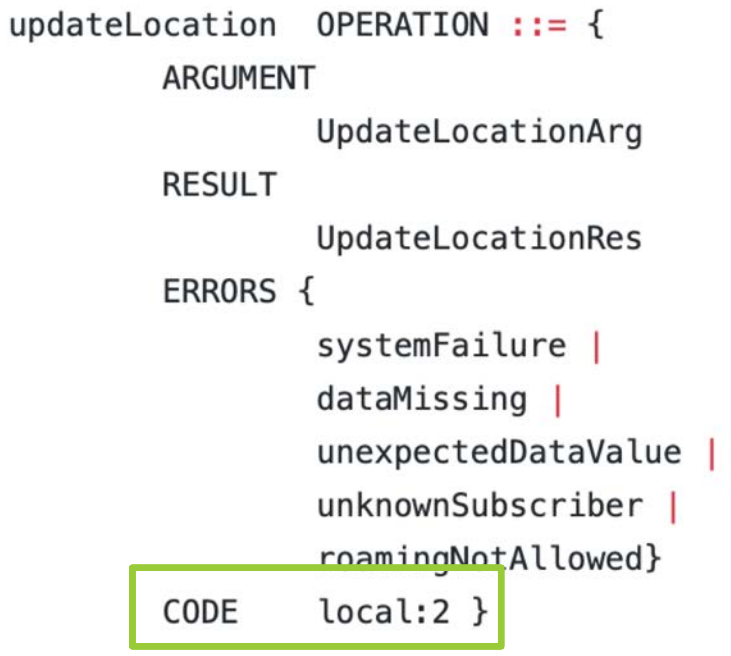
- At TCAP level
  - Retrieve Application Context Name, to identify a set of operations and a version



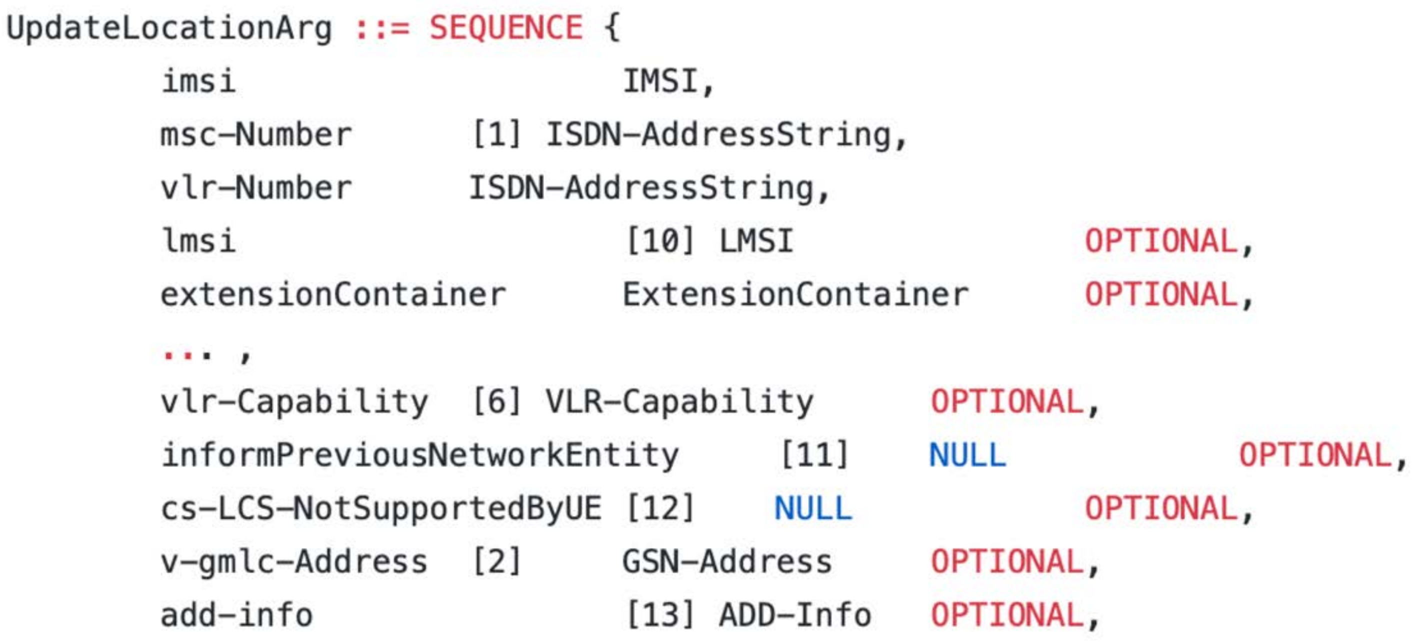
```
subscriberDataMngtContext-v3  OBJECT IDENTIFIER ::=
        {map-ac subscriberDataMngt(16) version3(3)}

tracingContext-v3  OBJECT IDENTIFIER ::=
        {map-ac tracing(17) version3(3)}
```

(SOME) TCAP/MAP TASKS FOR AN SS7 FIREWALL

- At MAP level
  - Retrieve local opcode, to identify the message in the set of operations
  - Parse and process message parts



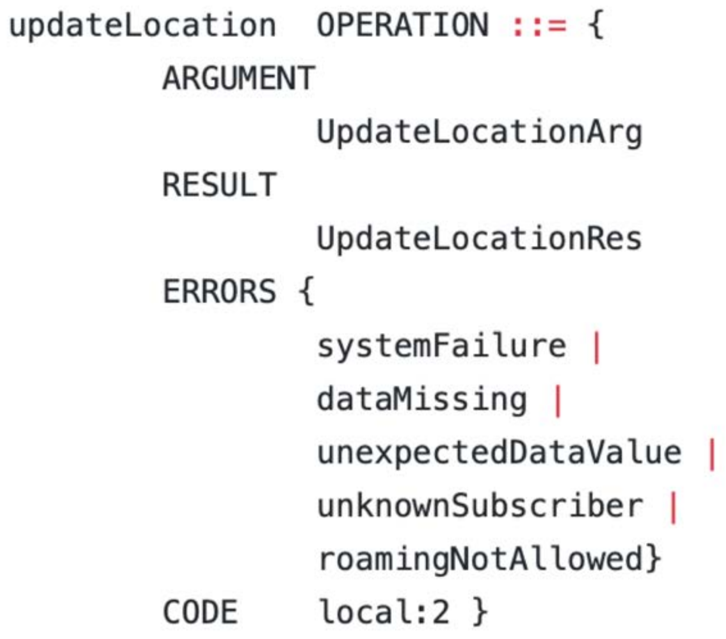
```
updateLocation  OPERATION ::= {
        ARGUMENT
                UpdateLocationArg
        RESULT
                UpdateLocationRes
        ERRORS {
                systemFailure |
                dataMissing |
                unexpectedDataValue |
                unknownSubscriber |
                roamingNotAllowed}
        CODE    local:2 }
```



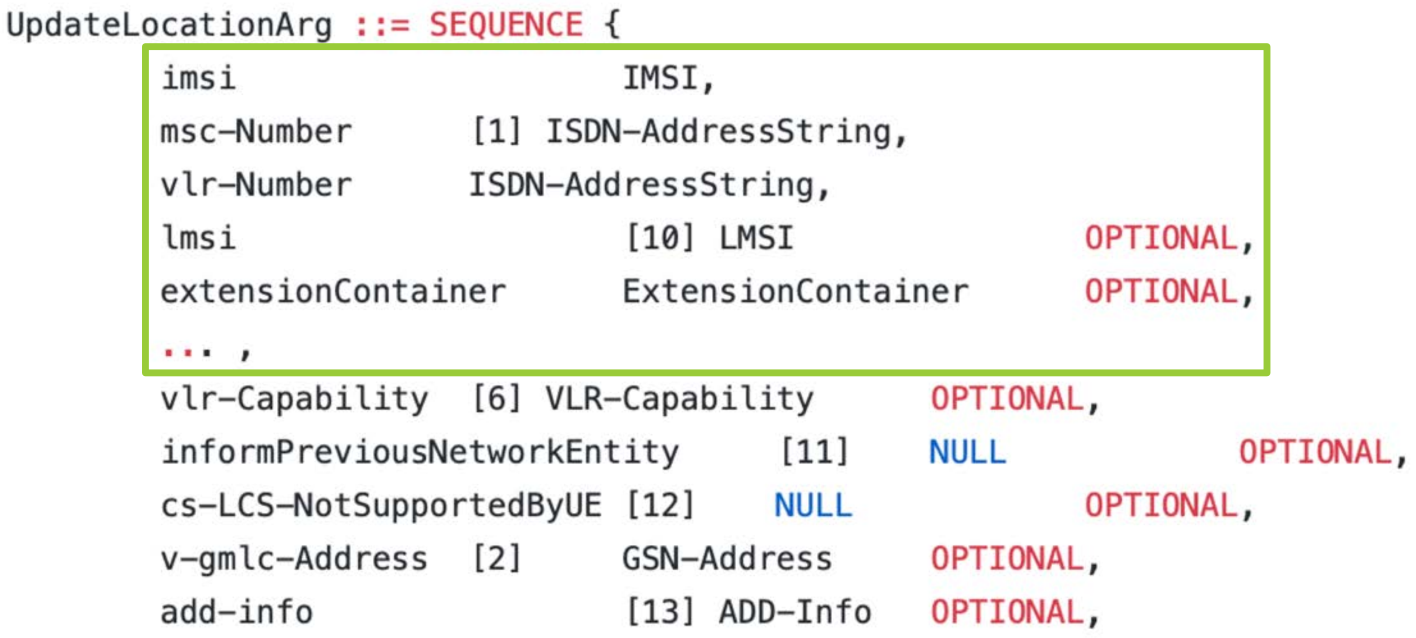
```
UpdateLocationArg ::= SEQUENCE {
        imsi                        IMSI,
        msc-Number        [1] ISDN-AddressString,
        vlr-Number        ISDN-AddressString,
        lmsi                        [10] LMSI                OPTIONAL,
        extensionContainer          ExtensionContainer       OPTIONAL,
        ... ,
        vlr-Capability  [6] VLR-Capability      OPTIONAL,
        informPreviousNetworkEntity     [11]    NULL                OPTIONAL,
        cs-LCS-NotSupportedByUE [12]    NULL                 OPTIONAL,
        v-gmlc-Address  [2]     GSN-Address     OPTIONAL,
        add-info                    [13] ADD-Info       OPTIONAL,
```

(SOME) TCAP/MAP TASKS FOR AN SS7 FIREWALL

- At MAP level
  - Retrieve local opcode, to identify the message in the set of operations
  - Parse and process message parts



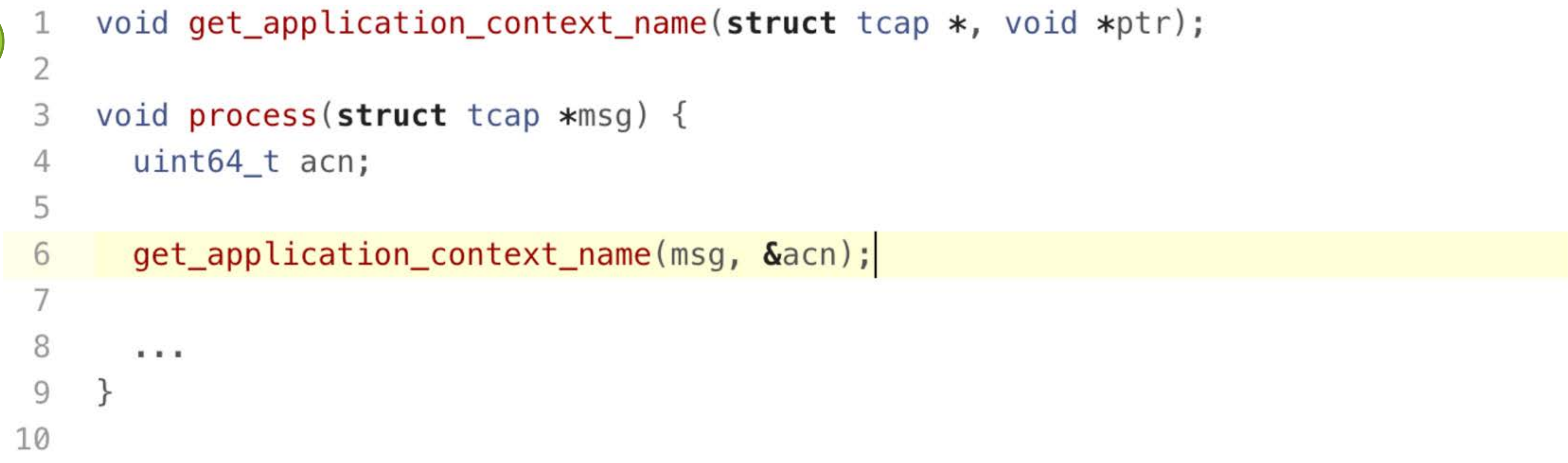
```
updateLocation  OPERATION ::= {
        ARGUMENT
                UpdateLocationArg
        RESULT
                UpdateLocationRes
        ERRORS {
                systemFailure |
                dataMissing |
                unexpectedDataValue |
                unknownSubscriber |
                roamingNotAllowed}
        CODE    local:2 }
```



```
UpdateLocationArg ::= SEQUENCE {
        imsi                        IMSI,
        msc-Number        [1] ISDN-AddressString,
        vlr-Number        ISDN-AddressString,
        lmsi                        [10] LMSI                     OPTIONAL,
        extensionContainer          ExtensionContainer            OPTIONAL,
        ... ,
        vlr-Capability    [6] VLR-Capability          OPTIONAL,
        informPreviousNetworkEntity       [11]        NULL                OPTIONAL,
        cs-LCS-NotSupportedByUE [12]      NULL                    OPTIONAL,
        v-gmlc-Address    [2]       GSN-Address       OPTIONAL,
        add-info                    [13] ADD-Info     OPTIONAL,
```

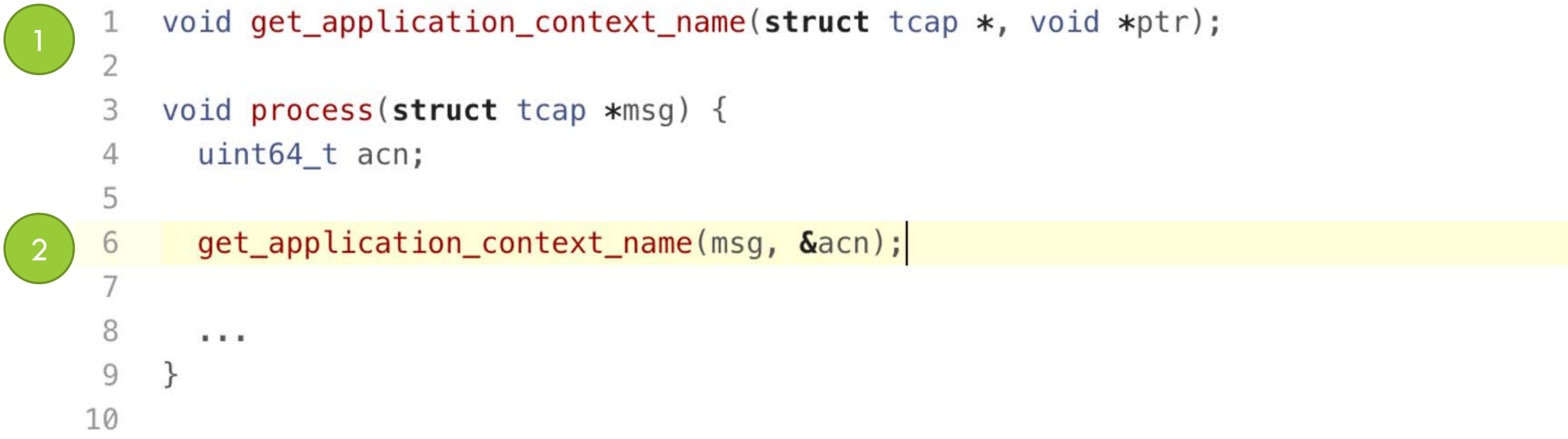
OVERFLOW STACK TO GAIN CODE EXECUTION CVE-2018-19563

1



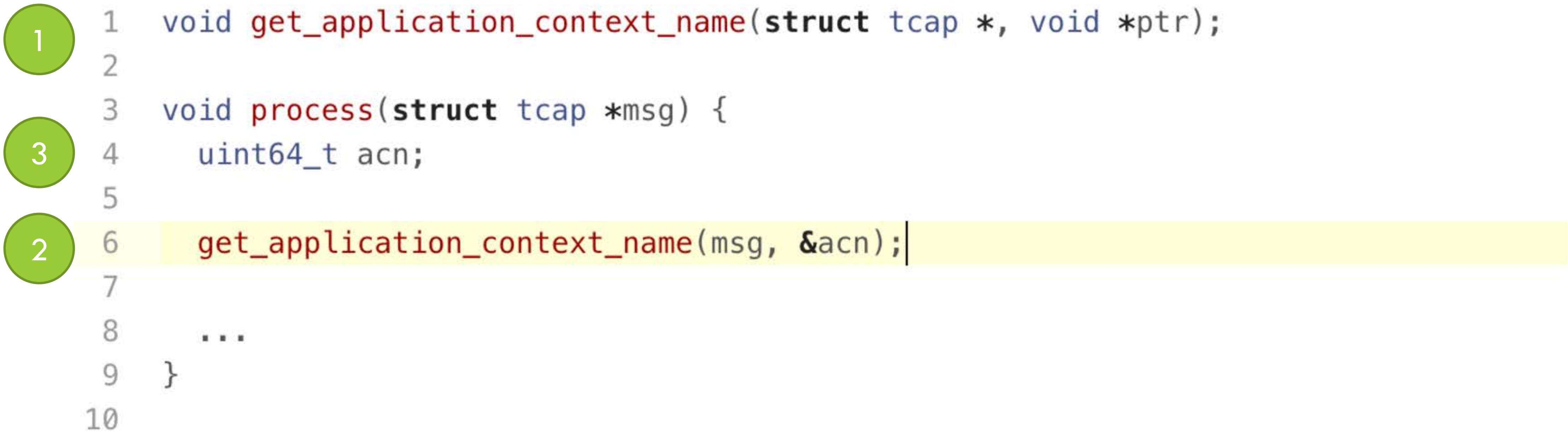
```c
void get_application_context_name(struct tcap *, void *ptr);

void process(struct tcap *msg) {
  uint64_t acn;

  get_application_context_name(msg, &acn);

  ...
}
```

OVERFLOW STACK
TO GAIN CODE EXECUTION
CVE-2018-19563



```c
void get_application_context_name(struct tcap *, void *ptr);

void process(struct tcap *msg) {
  uint64_t acn;

  get_application_context_name(msg, &acn);

  ...
}

```

OVERFLOW STACK
TO GAIN CODE EXECUTION
CVE-2018-19563

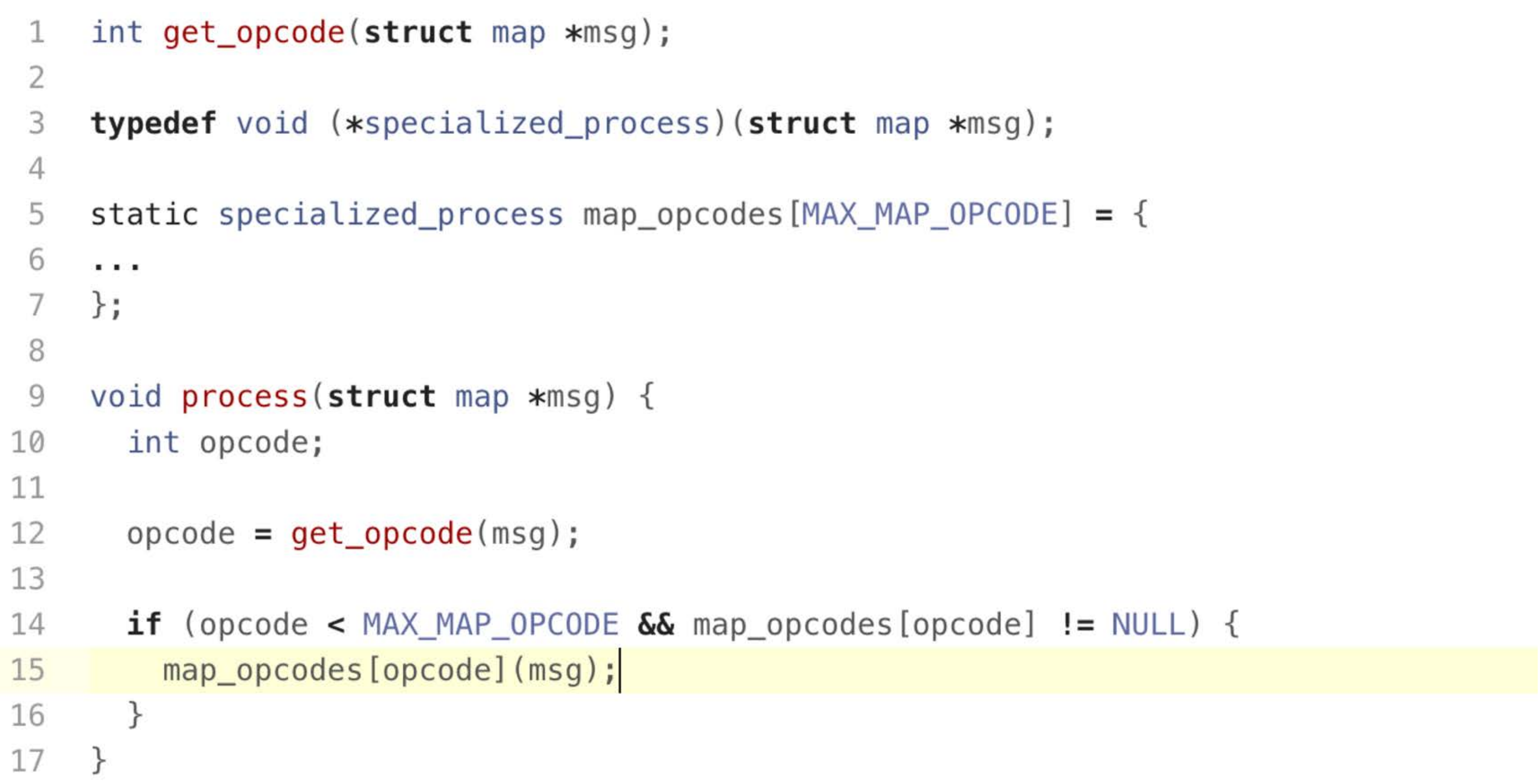


```c
void get_application_context_name(struct tcap *, void *ptr);

void process(struct tcap *msg) {
  uint64_t acn;

  get_application_context_name(msg, &acn);

  ...
}
```

Unless stated otherwise, ASN.1 primitive types can be *almost* arbitrary long

ABUSE SIGNEDNESS TO CALL ARBITRARY FUNCTION

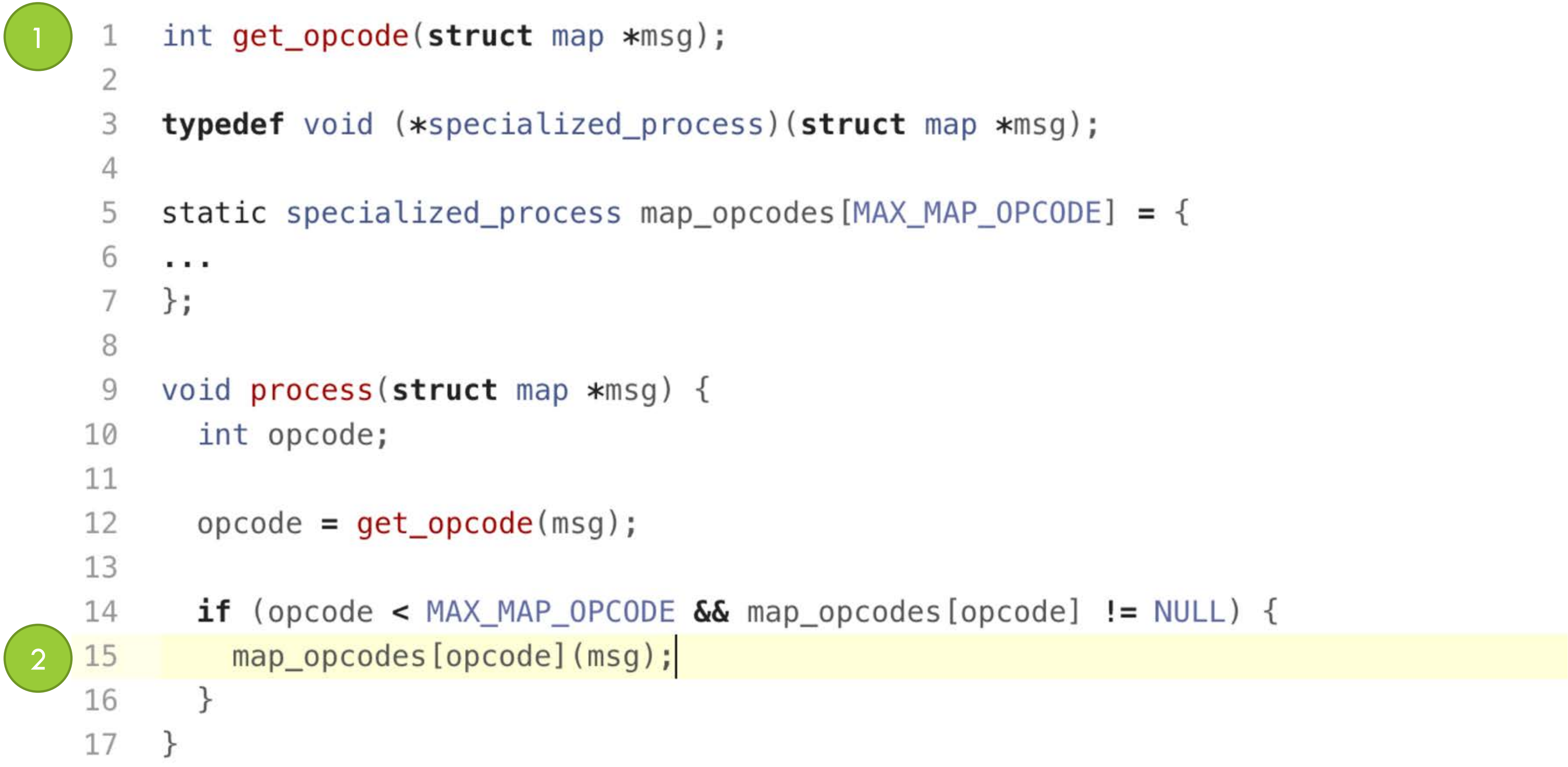
1



```c
int get_opcode(struct map *msg);

typedef void (*specialized_process)(struct map *msg);

static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
...
};

void process(struct map *msg) {
  int opcode;

  opcode = get_opcode(msg);

  if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
    map_opcodes[opcode](msg);
  }
}
```

ABUSE SIGNEDNESS

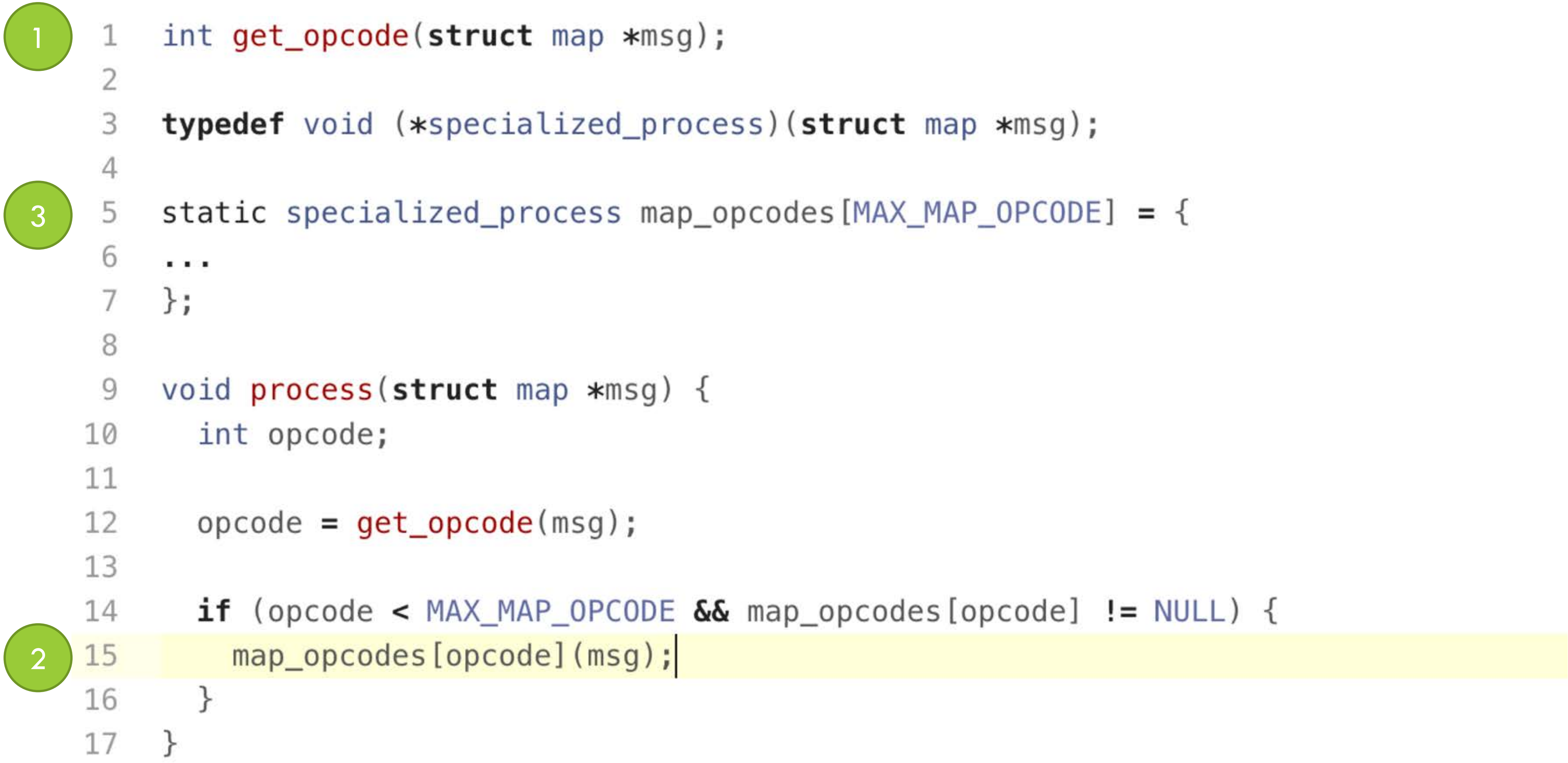
TO CALL ARBITRARY FUNCTION



```c
int get_opcode(struct map *msg);

typedef void (*specialized_process)(struct map *msg);

static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
...
};

void process(struct map *msg) {
  int opcode;

  opcode = get_opcode(msg);

  if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
    map_opcodes[opcode](msg);
  }
}
```

ABUSE SIGNEDNESS

TO CALL ARBITRARY FUNCTION



```c
int get_opcode(struct map *msg);

typedef void (*specialized_process)(struct map *msg);

static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
...
};

void process(struct map *msg) {
  int opcode;

  opcode = get_opcode(msg);

  if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
    map_opcodes[opcode](msg);
  }
}
```

ASN1 INTEGER primitive type is signed, and may be wider than actual machine width

- 5G will not make legacy networks disappear, protection mecanisms are required
- Vulnerabilites may sometimes be remotely exploited via SIGTRAN

- Legacy makes solution design clumsy
- Lack of hardening makes vulnerabilities easy to exploit

- Enhance hardening measures
  - Follow best practices for software robustness
  - Ensure mandatory access control
  - Fuzz efficiently every bit of software exposed to the wild

KEY TAKEAWAYS

THANK YOU

REMINDER: WEAKNESSES

- IP/SCTP/SCCP
  - Abuse segmentation & fragmentation to evade detection
- SCCP
  - CWE-125 Out-of-bounds memory access, causing a denial of service during parsing
  - CWE-789 Uncontrolled memory allocation during GT parsing
  - CWE-120 Buffer copy without checking size of input during reassembly
- TCAP/MAP
  - CWE-121 Stack-based buffer overflow in ASN1 primitive types
  - CWE-129 Improper validation of Array Index of MAP localOpcode
  - CWE-122 Heap-based buffer overflow in ASN1 constructed types