

MOBILE INTERCONNECT THREATS

How next-gen products may be
already outdated

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

[Matrices](#)[Tactics ▾](#)[Techniques ▾](#)[Groups](#)[Software](#)[Resources ▾](#)[Blog ↗](#)[Contribute](#)

ss7

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

@CWI_JohnnyE

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)

A MOVE TO DEFENSE



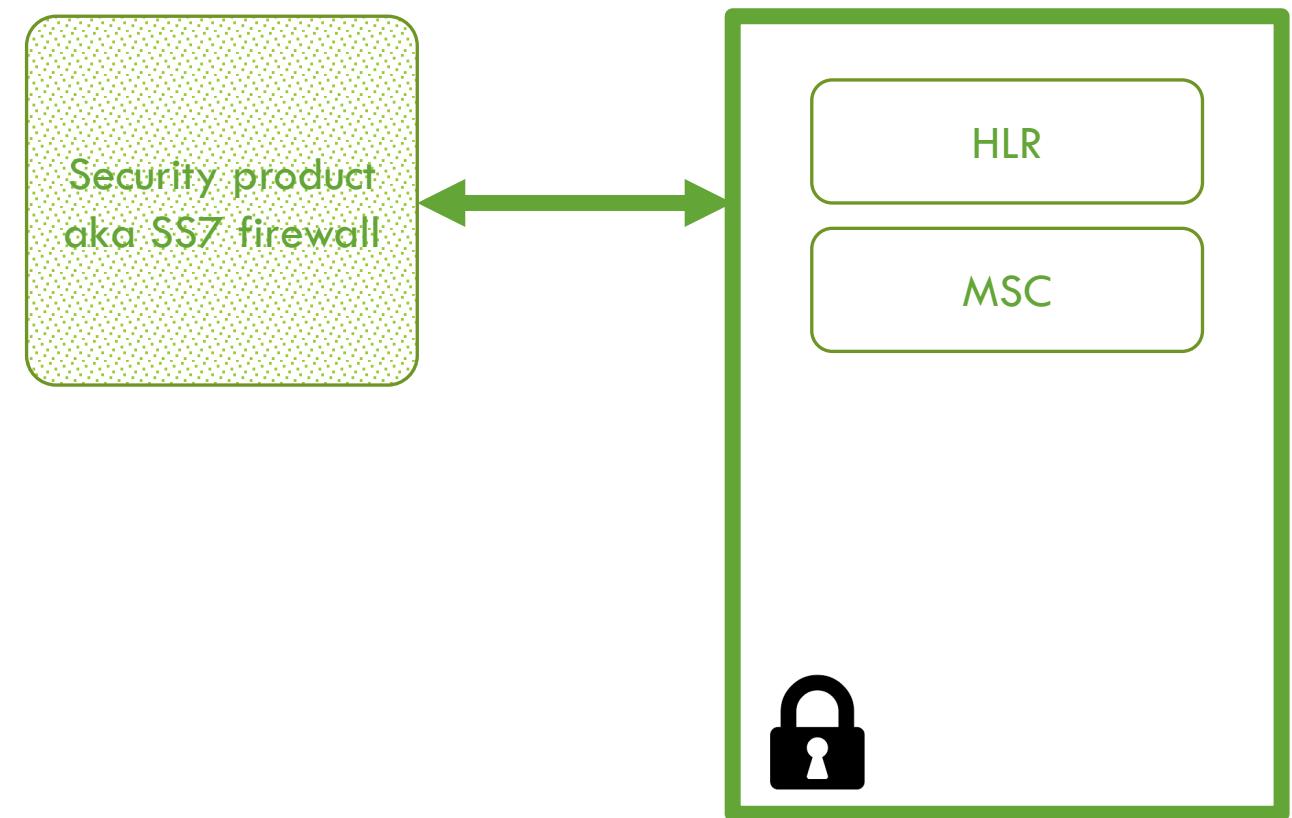
The image shows the cover of the GSMA Mobile Telecommunications Security Threat Landscape report from January 2019. The cover features a dark background with red glowing network nodes and lines, symbolizing a complex telecommunications network. In the top left corner is the GSMA logo, which consists of four red bars of increasing height followed by the acronym 'GSMA'. The main title 'Mobile Telecommunications Security Threat Landscape' is prominently displayed in large white font across the center. Below the title, the date 'January 2019' is written in a smaller white font.

Recommendations

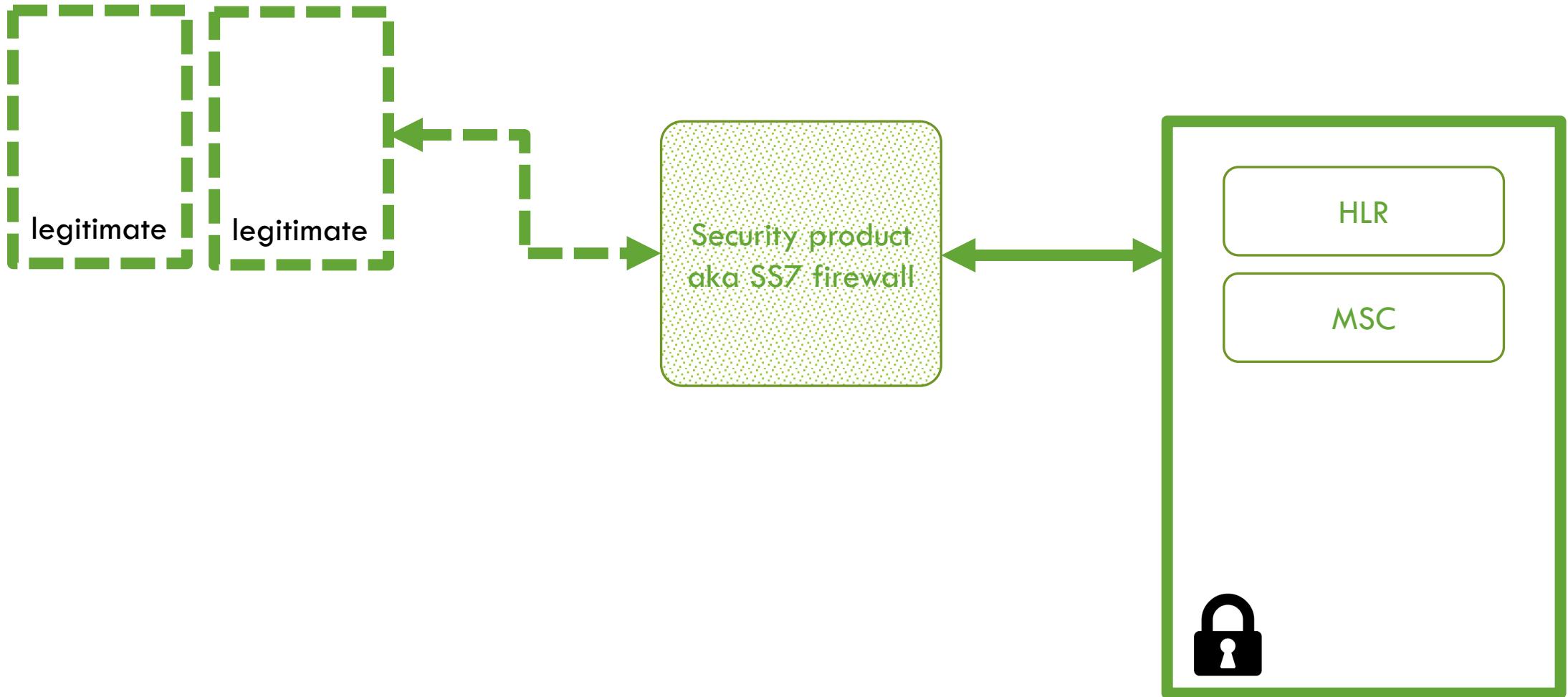
Current signalling protocols will remain within the industry for many years to come; as a result the GSMA recommend that operators implement compensating controls for these insecure protocols, specifically:

- Implement signalling controls outlined in the GSMA Fraud and Security Group³¹ (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.

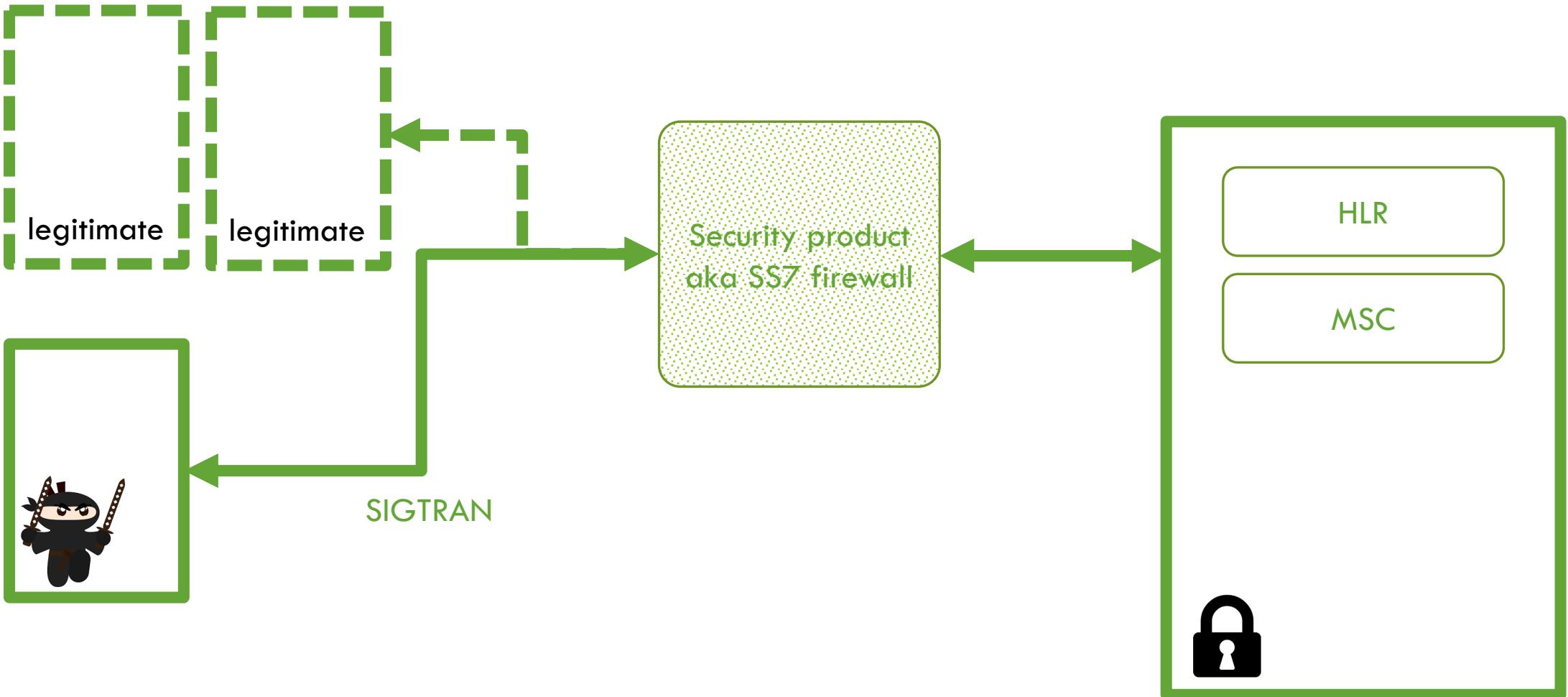
THREAT MODEL



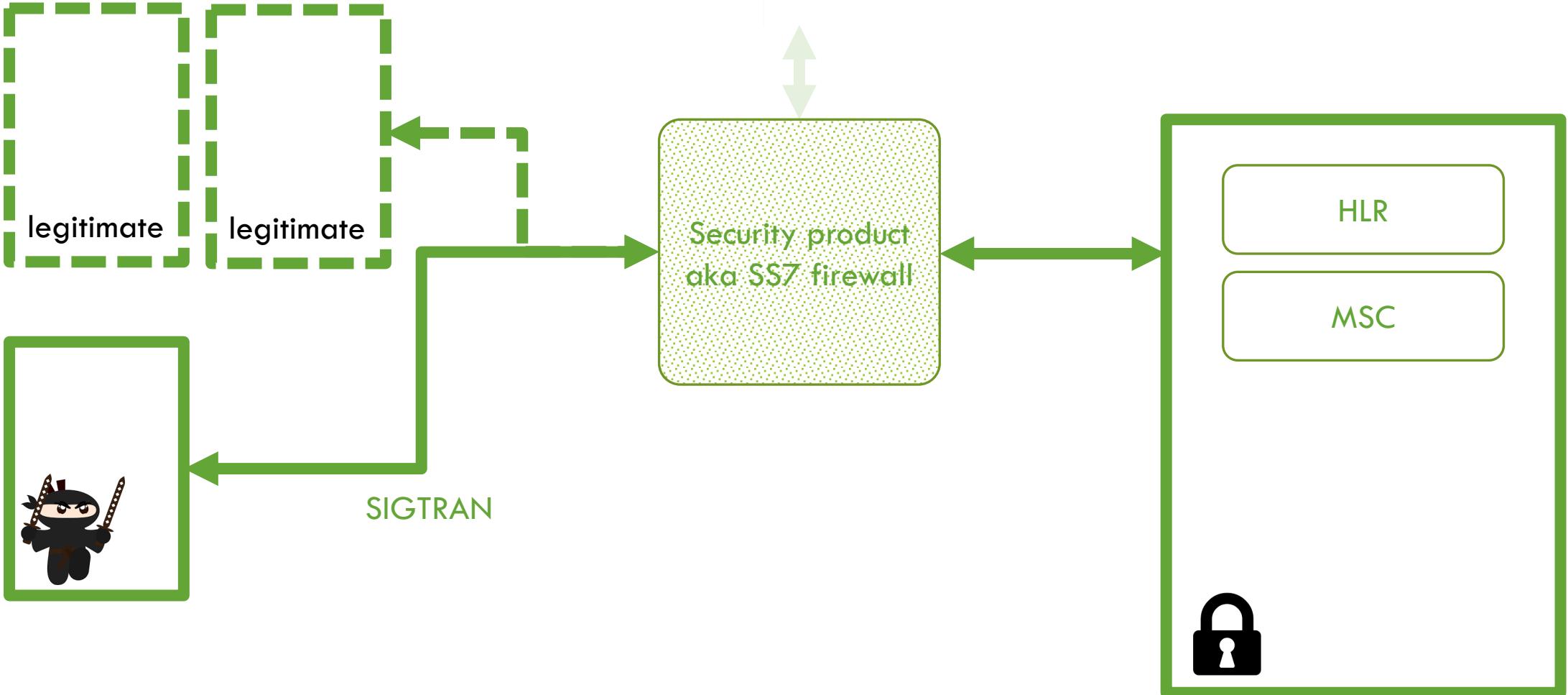
THREAT MODEL



THREAT MODEL



THREAT MODEL



Security product
aka SS7 firewall

SIGTRAN

OUR CONTRIBUTION |

Reverse engineering

Security product
aka SS7 firewall

SIGTRAN

OUR CONTRIBUTION |



Security product
aka SS7 firewall

SIGTRAN

Reverse engineering

Fuzzing



OUR CONTRIBUTION

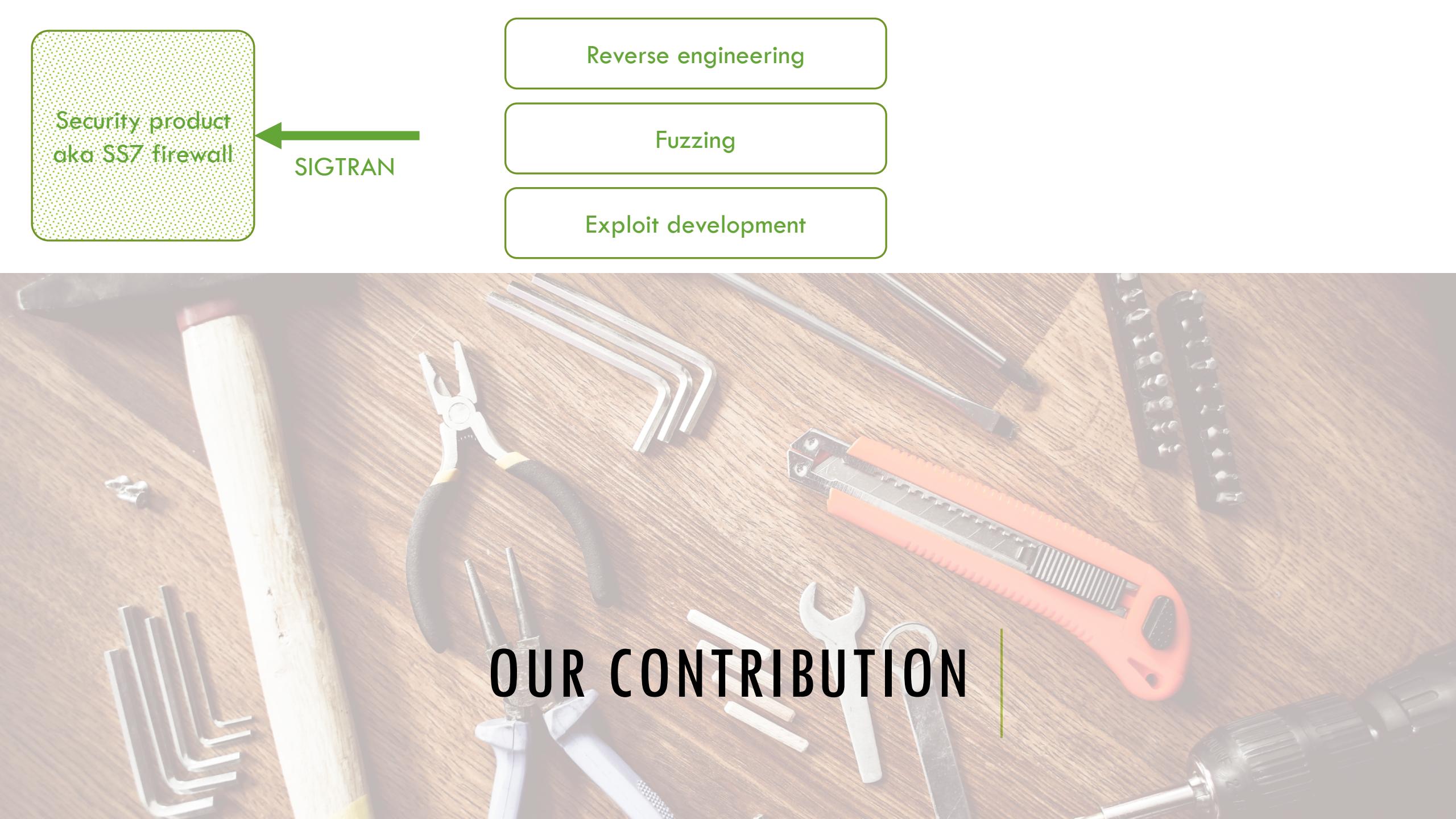
Security product
aka SS7 firewall

SIGTRAN

Reverse engineering

Fuzzing

Exploit development



OUR CONTRIBUTION |

Security product
aka SS7 firewall

SIGTRAN

Reverse engineering

Fuzzing

Exploit development

```
from pwnts7.ber import *
from pwnts7.protocols import *
from pwnts7.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=[
```

OUR CONTRIBUTION |



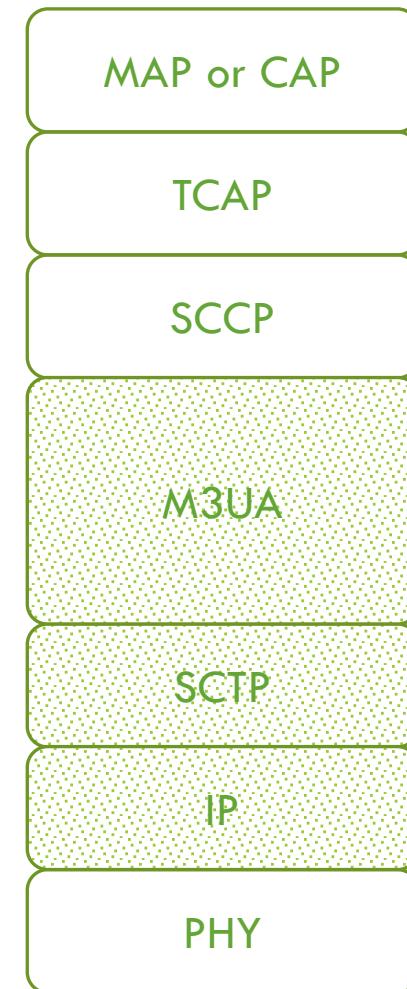
SS7 ON TOP OF IP TRANSPORT = SIGTRAN



SS7 ON TOP OF IP TRANSPORT = SIGTRAN



SS7 ON TOP OF IP TRANSPORT = SIGTRAN

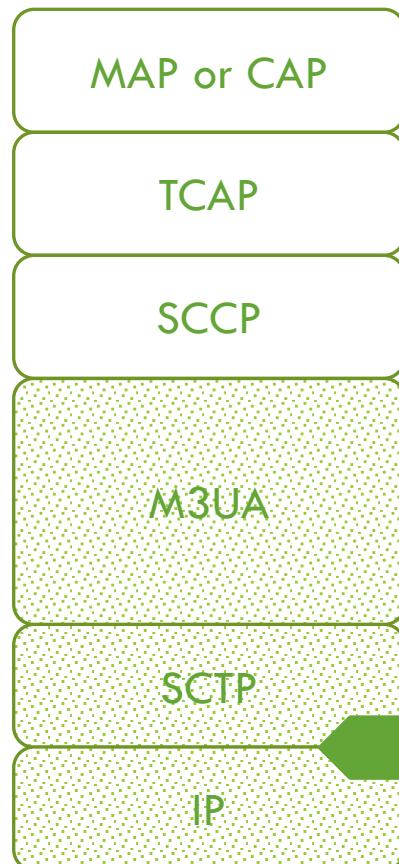


IP/SCTP

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 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

```
1 void process_ip(struct ip *msg) {  
2     if (IS_FRAGMENT(msg) || msg->proto != IPPROTO_SCTP) {  
3         return;  
4     }  
5  
6     process_sctp(msg->data);  
7 }
```

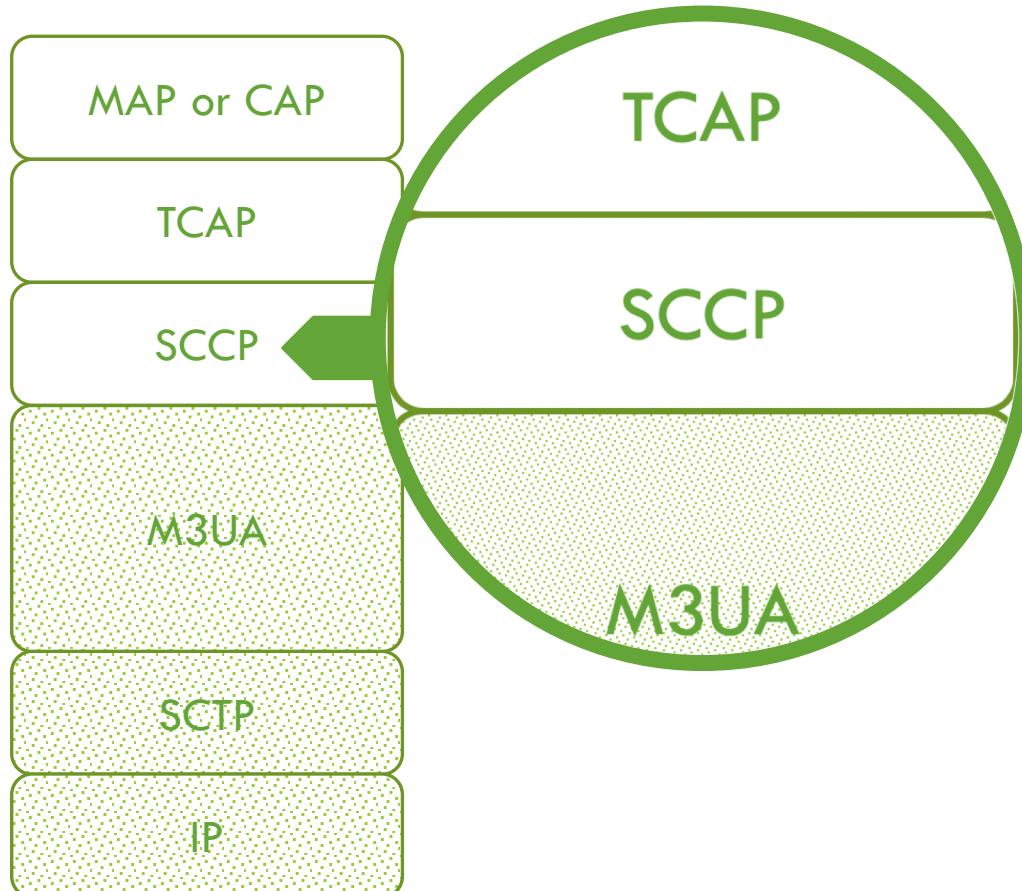
TRICK REASSEMBLY TO BYPASS DETECTION

```
1 void process_ip(struct ip *msg) {  
2     if (IS_FRAGMENT(msg) || msg->proto != IPPROTO_SCTP) {  
3         return;  
4     }  
5  
6     process_sctp(msg->data);  
7 }
```

```
1 void process_sctp(struct sctp *msg) {  
2     struct sctp_chunk *chunk;  
3  
4     foreach (chunk in msg->chunks) {  
5         if (chunk->type == DATA_CHUNK) {  
6             process_m3ua(chunk->data);  
7         }  
8     }  
9 }
```

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

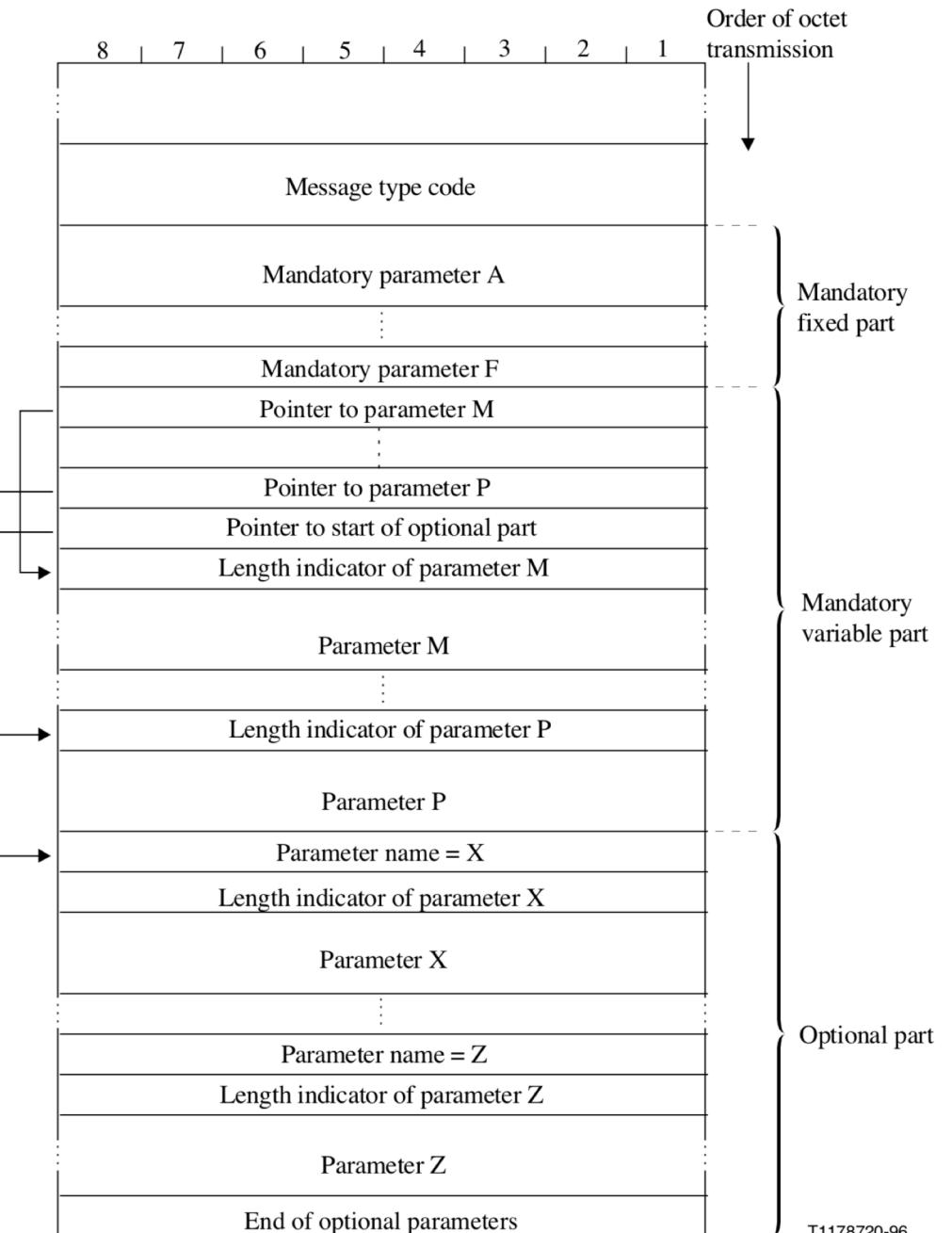
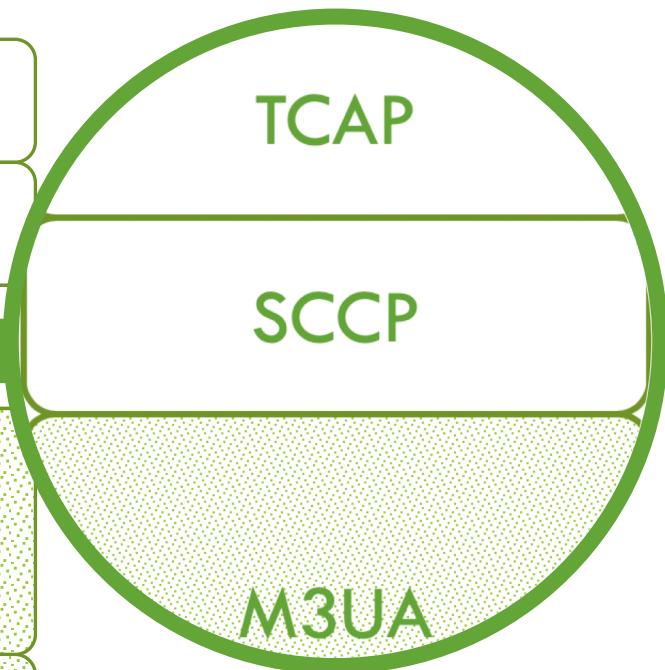
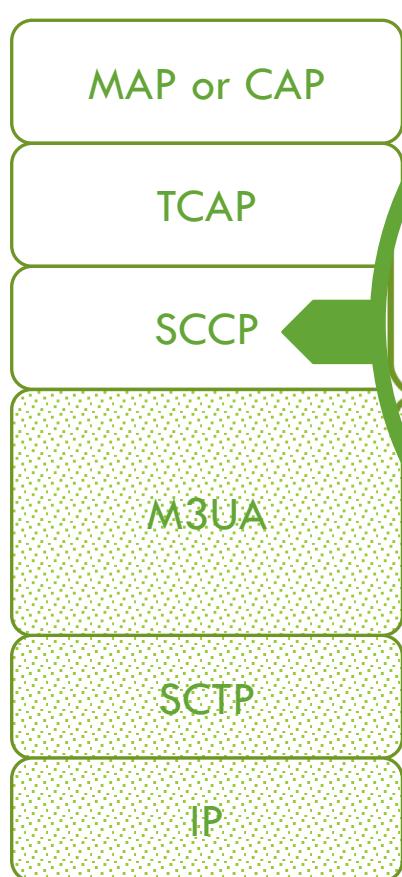


Figure 2/Q.713 – General SCCP message format

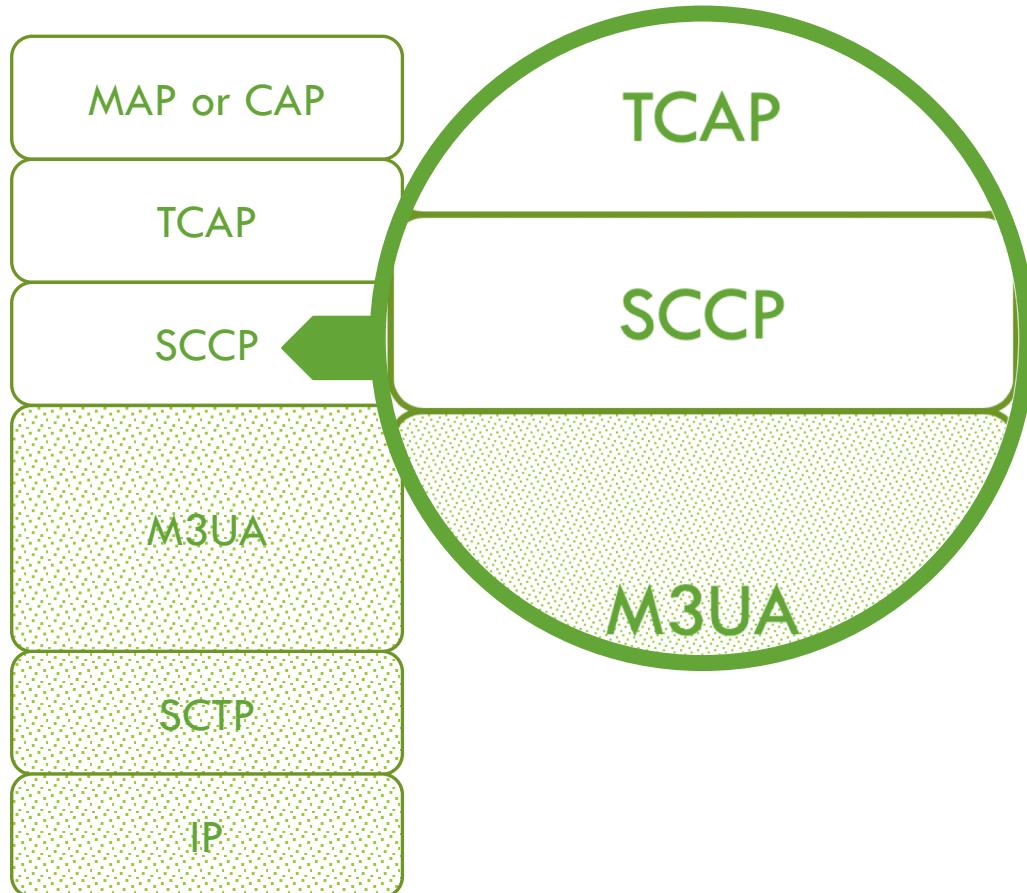
T1178720-96

TRICK FORWARD POINTERS TO DENY SERVICE

```
1 void process_sccp(const unsigned char *sccp, size_t size) {
2     unsigned char *called = NULL;
3     unsigned char *calling = NULL;
4
5     switch (sccp[0]) {
6         case 9:
7         case 10: /* UDT{,S} have fwd pointer on a single byte */
8             called = &sccp[2] + sccp[2];
9             break;
10        case 19:
11        case 20: /* LUDT{,S} have fwd pointer on short */
12            called = &sccp[2] + ntohs(*(unsigned short *)&sccp[2]);
13            break;
14    }
15
16    process_called(called);
17 }
```

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 encoded as 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, encoding 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 void process_udt(const unsigned char *ptr, size_t size) {
2     int gt_size;
3     const unsigned char *current;
4
5     /* ..., erroneous processing yields a negative gt_size */
6
7     process_calling(current, current + size, gt_size);
8 }
9
10 static void process_calling(const void *ptr, const char *end, int size) {
11     char digits[size];
12
13     process_gt(digits, ptr, max(ptr+size, end));
14 }
15
16 static void process_gt(char *digits, const void *ptr, const void *end) {
17     const char *c_ptr = ptr;
18
19     while (c_ptr != end) {
20         *digits = *c_ptr;
21         digits++;
22         c_ptr++;
23     }
```

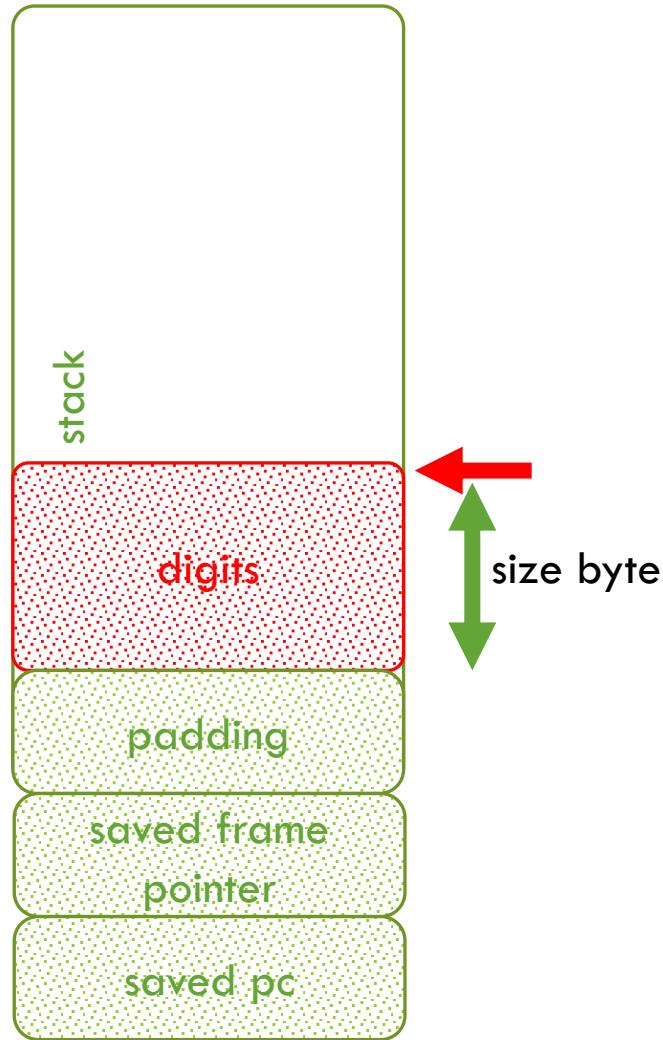
FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION

```
1 void process_udt(const unsigned char *ptr, size_t size) {
2     int gt_size;
3     const unsigned char *current;
4
5     /* ..., erroneous processing yields a negative gt_size */
6
7     process_calling(current, current + size, gt_size);
8 }
9
10 static void process_calling(const void *ptr, const char *end, int size) {
11     char digits[size];
12
13     process_gt(digits, ptr, max(ptr+size, end));
14 }
15
16 static void process_gt(char *digits, const void *ptr, const void *end) {
17     const char *c_ptr = ptr;
18
19     while (c_ptr != end) {
20         *digits = *c_ptr;
21         digits++;
22         c_ptr++;
23     }
```

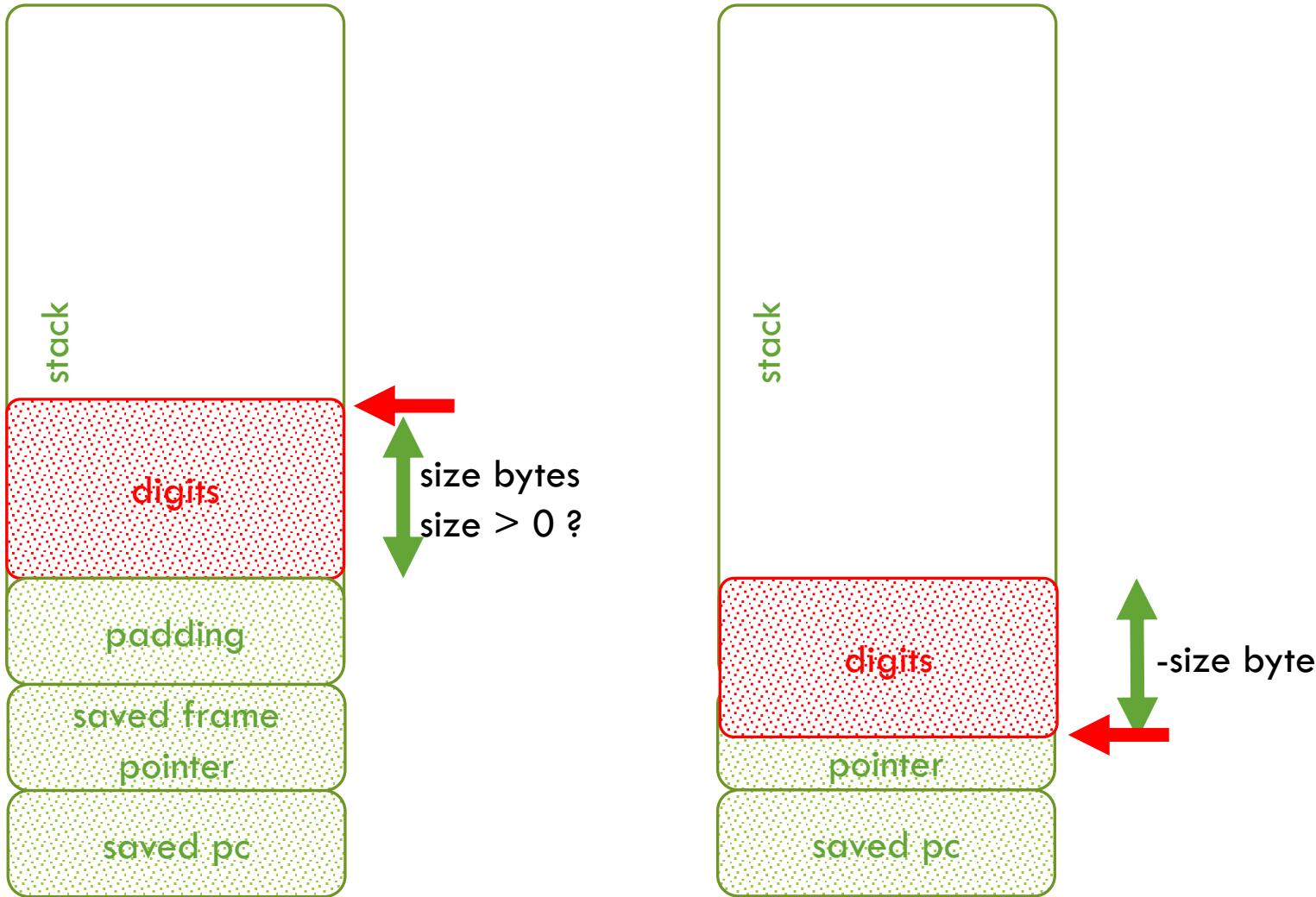
FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION

```
1 void process_udt(const unsigned char *ptr, size_t size) {
2     int gt_size;
3     const unsigned char *current;
4
5     /* ..., erroneous processing yields a negative gt_size */
6
7     process_calling(current, current + size, gt_size);
8 }
9
10 static void process_calling(const void *ptr, const char *end, int size) {
11     char digits[size];
12
13     process_gt(digits, ptr, max(ptr+size, end));
14 }
15
16 static void process_gt(char *digits, const void *ptr, const void *end) {
17     const char *c_ptr = ptr;
18
19     while (c_ptr != end) {
20         *digits = *c_ptr;
21         digits++;
22         c_ptr++;
23     }
```

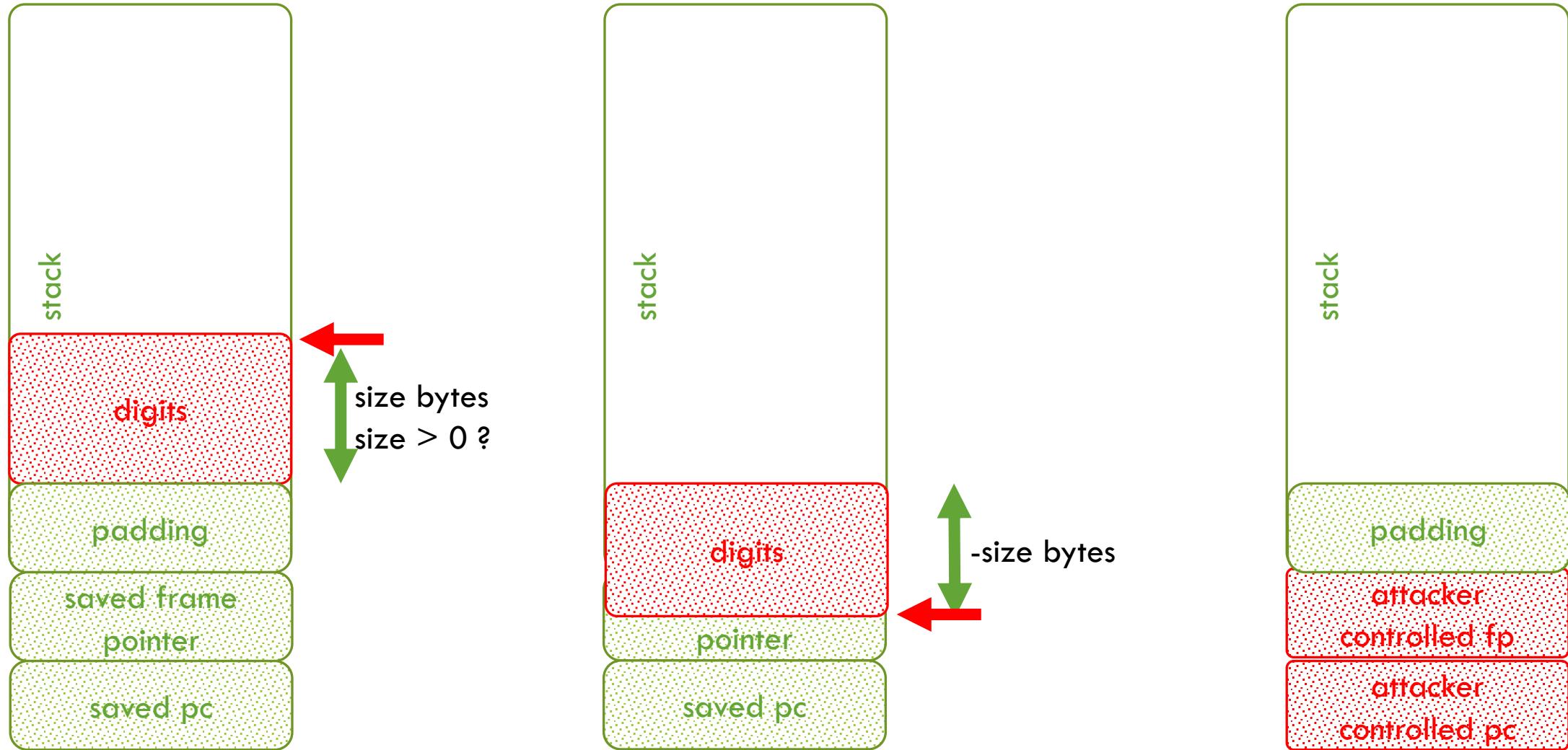
FOOLING GLOBAL TITLES TO GAIN CODE EXECUTION



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

```
1 static size_t reassembled = 0;
2 static unsigned char big_buffer[8192];
3
4 static unsigned char smaller_buffer[1024];
5
6
7
8
9
10
11 void process_xudt(struct xudt *msg) {
12     if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
13         memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
14         reassembled += msg->fragment_size;
15     }
16     if (msg->M == 0) {
17         memcpy(smaller_buffer, big_buffer, reassembled);
18         reassembled = 0;
19     }
20 }
```

--

OVERFLOW REASSEMBLY TO GAIN CODE EXECUTION

```
1 static size_t reassembled = 0;
2 static unsigned char big_buffer[8192];
3
4 static unsigned char smaller_buffer[1024];
5
6
7
8
9
10
11 void process_xudt(struct xudt *msg) {
12     if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
13         memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
14         reassembled += msg->fragment_size;
15     }
16     if (msg->M == 0) {
17         memcpy(smaller_buffer, big_buffer, reassembled); 2
18         reassembled = 0;
19     }
20 }
```

OVERFLOW REASSEMBLY TO GAIN CODE EXECUTION

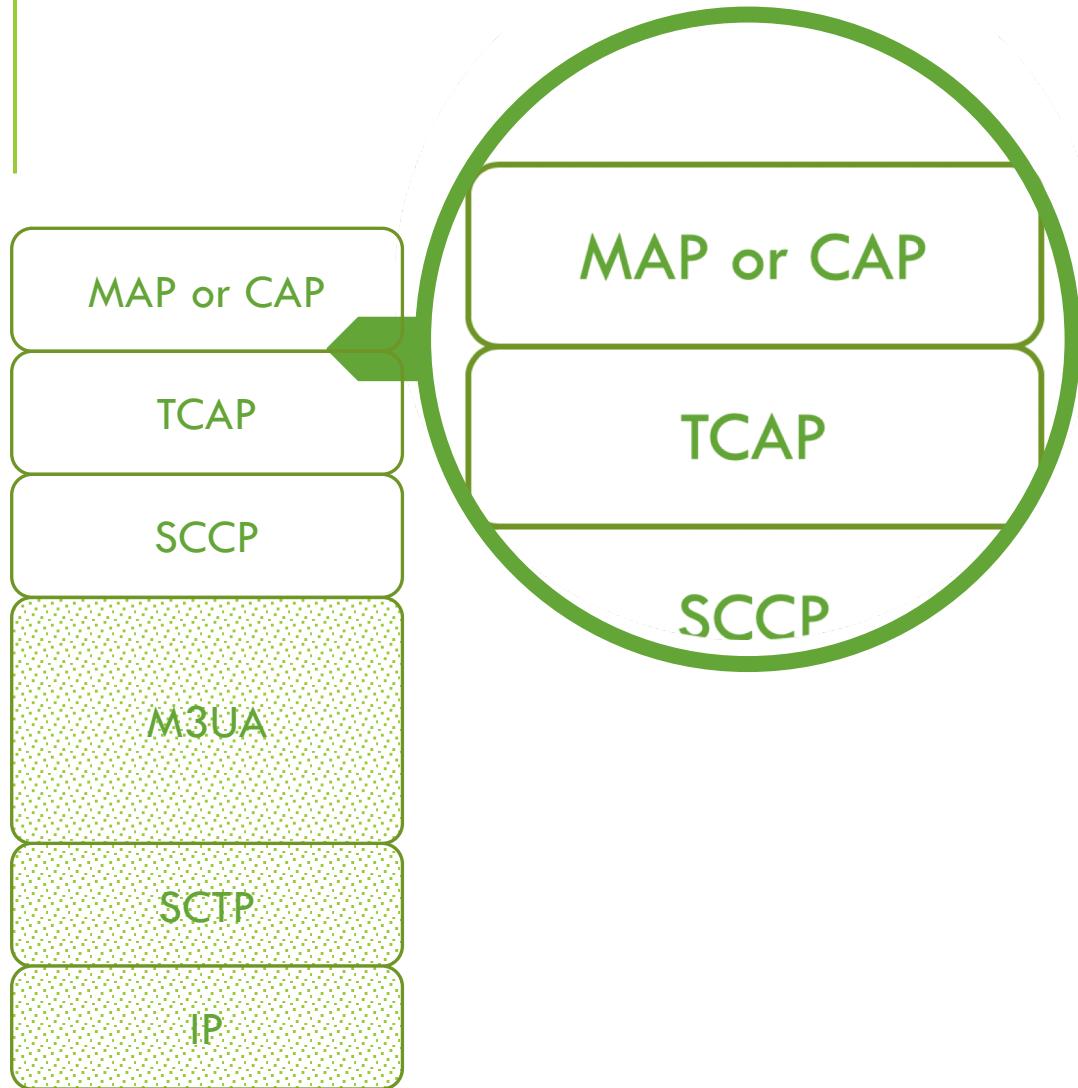
```
1 static size_t reassembled = 0;
2 static unsigned char big_buffer[8192];
3
4 static unsigned char smaller_buffer[1024];
5
6
7
8
9
10
11 void process_xudt(struct xudt *msg) {
12     if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
13         memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
14         reassembled += msg->fragment_size;
15     }
16     if (msg->M == 0) {
17         memcpy(smaller_buffer, big_buffer, reassembled);
18         reassembled = 0;
19     }
20 }
```

OVERFLOW REASSEMBLY TO GAIN CODE EXECUTION

```
1 static size_t reassembled = 0;
2 static unsigned char big_buffer[8192];
3
4 static unsigned char smaller_buffer[1024];
5
6 static struct {
7     const void *head;
8     const void *tail;
9 } fragments;
10
11 void process_xudt(struct xudt *msg) {
12     if (reassembled + msg->fragment_size <= sizeof(big_buffer)) {
13         memcpy(&big_buffer[reassembled], msg->fragment, msg->fragment_size);
14         reassembled += msg->fragment_size;
15     }
16     if (msg->M == 0) {
17         memcpy(smaller_buffer, big_buffer, reassembled);|
18         reassembled = 0;
19     }
20 }
```

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



asn1 compiler

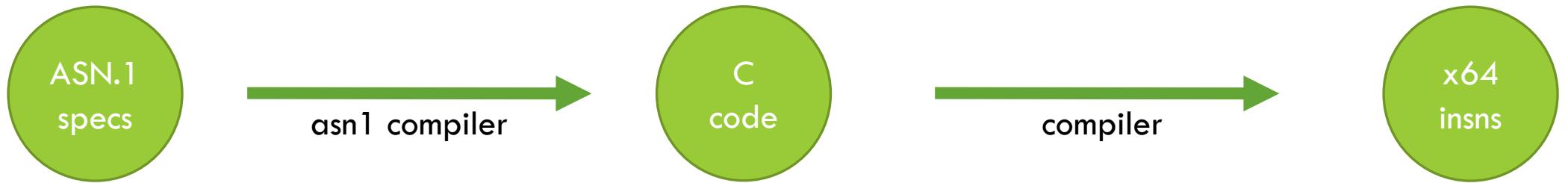


```
10042 static const ber_sequence_t RoutingInfoForSMArg_sequence[] = {  
10043     { BER_CLASS_CON, 0, BER_FLAGS_IMPLTAG, dissect_msisdn_impl },  
10044     { BER_CLASS_CON, 1, BER_FLAGS_IMPLTAG, dissect_sm_RP_PRI_impl },  
10045     { BER_CLASS_CON, 2, BER_FLAGS_IMPLTAG, dissect_serviceCentreAddress_impl  
     . },  
10046     { BER_CLASS_CON, 6, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,  
     . dissect_extensionContainer_impl },  
10047     { BER_CLASS_CON, 7, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,  
     . dissect_gprsSupportIndicator_impl },  
10048     { BER_CLASS_CON, 8, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,  
     . dissect_sm_RP_MTI_impl },  
10049     { BER_CLASS_CON, 9, BER_FLAGS_OPTIONAL|BER_FLAGS_IMPLTAG,  
     . dissect_sm_RP_SMEA_impl },  
10050     { 0, 0, 0, NULL }  
10051 };  
10052  
10053 static int  
10054 dissect_gsm_map_RoutingInfoForSMArg(gboolean implicit_tag _U_, tvbuff_t  
    . *tvb, int offset, packet_info *pinfo _U_, proto_tree *tree, int hf_index  
    . _U_) {  
10055     offset = dissect_ber_sequence(implicit_tag, pinfo, tree, tvb, offset,  
10056                                         RoutingInfoForSMArg_sequence, hf_index,  
     . ett_gsm_map_RoutingInfoForSMArg);  
10057  
10058     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 void get_application_context_name(struct tcap *, void *ptr);  
2  
3 void process(struct tcap *msg) {  
4     uint64_t acn;  
5  
6     get_application_context_name(msg, &acn);|  
7  
8     ...  
9 }  
10
```

OVERFLOW STACK

TO GAIN CODE EXECUTION

CVE-2018-19563

```
1 void get_application_context_name(struct tcap *, void *ptr);  
2  
3 void process(struct tcap *msg) {  
4     uint64_t acn;  
5  
6     get_application_context_name(msg, &acn);|  
7  
8     ...  
9 }  
10
```

OVERFLOW STACK

TO GAIN CODE EXECUTION
CVE-2018-19563

```
1 void get_application_context_name(struct tcap *, void *ptr);  
2  
3 void process(struct tcap *msg) {  
4     uint64_t acn;  
5  
6     get_application_context_name(msg, &acn);|  
7  
8     ...  
9 }  
10
```

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

ABUSE SIGNEDNESS TO CALL ARBITRARY FUNCTION

1

```
1 int get_opcode(struct map *msg);
2
3 typedef void (*specialized_process)(struct map *msg);
4
5 static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
6 ...
7 };
8
9 void process(struct map *msg) {
10    int opcode;
11
12    opcode = get_opcode(msg);
13
14    if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
15        map_opcodes[opcode](msg);
16    }
17 }
```

ABUSE SIGNEDNESS TO CALL ARBITRARY FUNCTION

```
1 int get_opcode(struct map *msg);
2
3 typedef void (*specialized_process)(struct map *msg);
4
5 static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
6 ...
7 };
8
9 void process(struct map *msg) {
10    int opcode;
11
12    opcode = get_opcode(msg);
13
14    if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
15        map_opcodes[opcode](msg);
16    }
17 }
```

ABUSE SIGNEDNESS TO CALL ARBITRARY FUNCTION

```
1 int get_opcode(struct map *msg);
2
3 typedef void (*specialized_process)(struct map *msg);
4
5 static specialized_process map_opcodes[MAX_MAP_OPCODE] = {
6 ...
7 };
8
9 void process(struct map *msg) {
10    int opcode;
11
12    opcode = get_opcode(msg);
13
14    if (opcode < MAX_MAP_OPCODE && map_opcodes[opcode] != NULL) {
15        map_opcodes[opcode](msg);
16    }
17 }
```

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

OVERFLOW HEAP TO GAIN CODE EXECUTION

```
1 struct some_map_msg {  
2     ...  
3     struct hlr hlrs[42];  
4 }  
5  
6 void process(struct map *msg) {  
7     struct some_map_msg *mmsg;  
8     size_t i;  
9  
10    mmsg = malloc(sizeof(*mmsg));  
11  
12    i = 0;  
13    foreach (hlr in msg) {  
14        mmsg->hlrs[i] = hlr;  
15        i++;  
16    }  
17  
18    ...  
19 }  
20
```

OVERFLOW HEAP TO GAIN CODE EXECUTION

```
1 struct some_map_msg {  
2     ...  
3     struct hlr hlrs[42];  
4 }  
5  
6 void process(struct map *msg) {  
7     struct some_map_msg *mmsg;  
8     size_t i;  
9  
10    mmsg = malloc(sizeof(*mmsg));  
11  
12    i = 0;  
13    foreach (hlr in msg) {  
14        mmsg->hlrs[i] = hlr;  
15        i++;  
16    }  
17  
18    ...  
19 }  
20
```

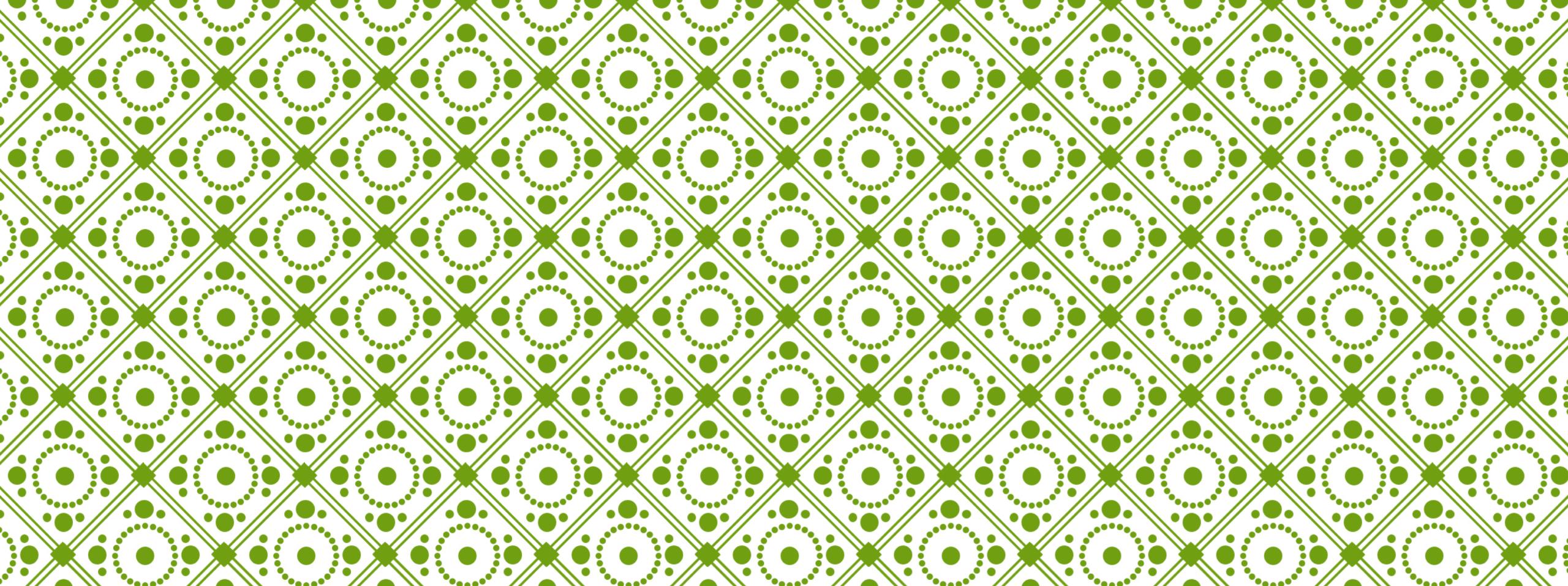
OVERFLOW HEAP TO GAIN CODE EXECUTION

```
1 struct some_map_msg {  
2     ...  
3     struct hlr hlrs[42];  
4 }  
5  
6 void process(struct map *msg) {  
7     struct some_map_msg *mmsg;  
8     size_t i;  
9  
10    mmsg = malloc(sizeof(*mmsg));  
11  
12    i = 0;  
13    foreach (hlr in msg) {  
14        mmsg->hlrs[i] = hlr;  
15        i++;  
16    }  
17  
18    ...  
19 }  
20
```

ASN.1 constructed types can contain any number of objects

- 5G will not make legacy networks disappear, protection mechanisms are required
- Vulnerabilities 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