**Research Background**

The number of cellular baseband players in the top-tier smartphone market is fairly small: Qualcomm, HiSilicon, Samsung, MediaTek, and Intel. With the exception of Intel, most of these manufacturers have been in the spotlight of relatively recent public security research in the mobile space (albeit more is needed and [coming](https://docs.google.com/presentation/d/19A1JWyOTueZvD8AksqCxtxriNJJgj0vPdq3cNTwndf4/edit#slide=id.g35506ef05e_0_0)!). Intel’s market share is relatively [small](https://www.barrons.com/articles/qualcomm-still-standing-tall-in-baseband-despite-intel-says-wells-fargo-1490959969) counting by the number of flagship devices they could score. The bulk of the market is dominated by Qualcomm solutions. On the flip side Intel modem solutions are among high-value targets with the integration of its [XMM7360](https://ark.intel.com/products/66649/Intel-XMM-7360) solutions into Apple iPhone devices since the [introduction](http://www.techinsights.com/about-techinsights/overview/blog/apple-iphone-7-teardown/) of the iPhone 7 in 2016. It has since been used in every new generation of iPhone devices. Because of a different feature set and different regional requirements, iPhones have been available in two flavors. While carriers using CDMA have traditionally used Qualcomm-powered baseband stacks, a large share of the remaining market runs on the Intel solution. With the introduction of CDMA in the upcoming Intel [XMM7560](https://www.intel.com/content/www/us/en/wireless-products/mobile-communications/xmm-7560-brief.html) platform, the developments in this space will be even more interesting to observe.

With this research we hope to make inroads into public research on vulnerabilities in Intel’s cellular platform. We also hope that this helps other researchers and curious folks to dive into the larger research area of cellular basebands. All of our work is based on manual reverse engineering of the involved XMM ARM (*Cortex-A5*) components that can be extracted from iPhone firmware downloads. To the best of our knowledge, this also is the first public write-up of a non-trivial cellular baseband memory corruption vulnerability. The issues affect all recent iPhone devices powered by the Intel XMM solution starting with the iPhone 7 until iOS 11.2.6. Apple [addressed](https://support.apple.com/kb/HT208693) the issues in iOS 11.3.

Going forward, we will use the issues to highlight various aspects around cellular security, its complexity, historic developments of the underlying specifications, and the challenges associated with implementing protocols, conducting research, and exploiting vulnerabilities. Some of these aspects have been widely mentioned in informal conversations, but are often difficult to understand due to a lack of examples. We hope that this can serve as an interesting sample in the very small pool of publicly discussed cellular baseband vulnerabilities.

If you want to skip the historic abstract about the standardization side and just want to get straight to the vulnerabilities, click [here](https://comsecuris.com/blog/posts/theres_life_in_the_old_dog_yet_tearing_new_holes_into_inteliphone_cellular_modems/#CVE-2018-4148).

**Warning Systems**

On January 13th 2018, the Hawaiian population was alarmed about an [incoming missile strike](https://www.washingtonpost.com/news/post-nation/wp/2018/01/13/hawaii-residents-get-ballistic-missile-threat-messages/) via a broadcast transmission issued over TV, cellular networks, and radio stations. Shortly afterwards, it became clear that this warning was falsely issued. By coincidence, we were also looking into emergency alert implementations within a popular cellular baseband implementation. Specifically, Intel’s cellular modem XMM7360 as used in iPhone 7 devices. Its successor XMM7460 is very similar and used in newer iPhone devices.



**Hawaiian Missile Alert in January 2018**

As part of this research, we identified a fairly interesting vulnerability in precisely the handling of alert messages. We will walk you through the elements of such warning and alert systems, its specifications, and finally a vulnerabilities.

Before diving deeply into the respective specifications and attack surface, we want to highlight the historic development of some of the underlying ideas over the years. For this we we will make a quick excursion into the world of broadcast information in cellular networks and the development of warning systems.

**Cell Broadcast Background**

Cellular communication channels usually support the concept of broadcast and non-broadcast channels. Broadcast channels are as the name already suggests, radio channels that are used for one-to-many communication, i.e. a radio resource that can be monitored by arbitrary devices camping on the network. This concept can be found in all relevant *Radio Access Technologies* (RAT) used for communication around the globe: CDMA, GSM, UMTS, and LTE.

These broadcast channels are used to transmit all kinds of information. Common examples are announcing network identities (e.g. name) and a list of supported carrier frequencies. When a mobile phone wants to roam into a cell, this information is important. There are multiple types of broadcast channels that are used differently by mobile devices. They are transmitted on downlink frequencies at all times. The way this concept is realized for example in GSM is by dividing the physical downlink frequency into timeslots by means of *Time Division Multiple Access* ([TDMA](https://en.wikipedia.org/wiki/Time-division_multiple_access)). TDMA allows creating logical channels at specific time slots within a frame transmitted on a physical frequency (in GSM e.g. 51-multiframes).

These logical channels can be further split into sub-slots, which carry information for specific RAT features such as [cell broadcast](https://en.wikipedia.org/wiki/Cell_Broadcast), which is implemented through the *Cell Broadcast Channel* (CBCH). The CBCH maps to a sub-slot on so-called [*Stand-alone Dedicated Control Channels*](https://en.wikipedia.org/wiki/Control_channel) (SDCCH), but is generally counted as part of the so-called *Common Control Channels* (CCH).

On top of this cell broadcast channel, the [ETSI](http://etsi.org/) GSM committee specified a **Cell Broadcast Service** (CBS), which was first demonstrated to the public in 1997. It is now part of 2G, 3G, and 4G technologies. The technical details can be found in the original GSM specification [ETSI TS GSM 03.41](http://www.etsi.org/deliver/etsi_gts/03/0341/05.01.00_60/gsmts_0341v050100p.pdf).

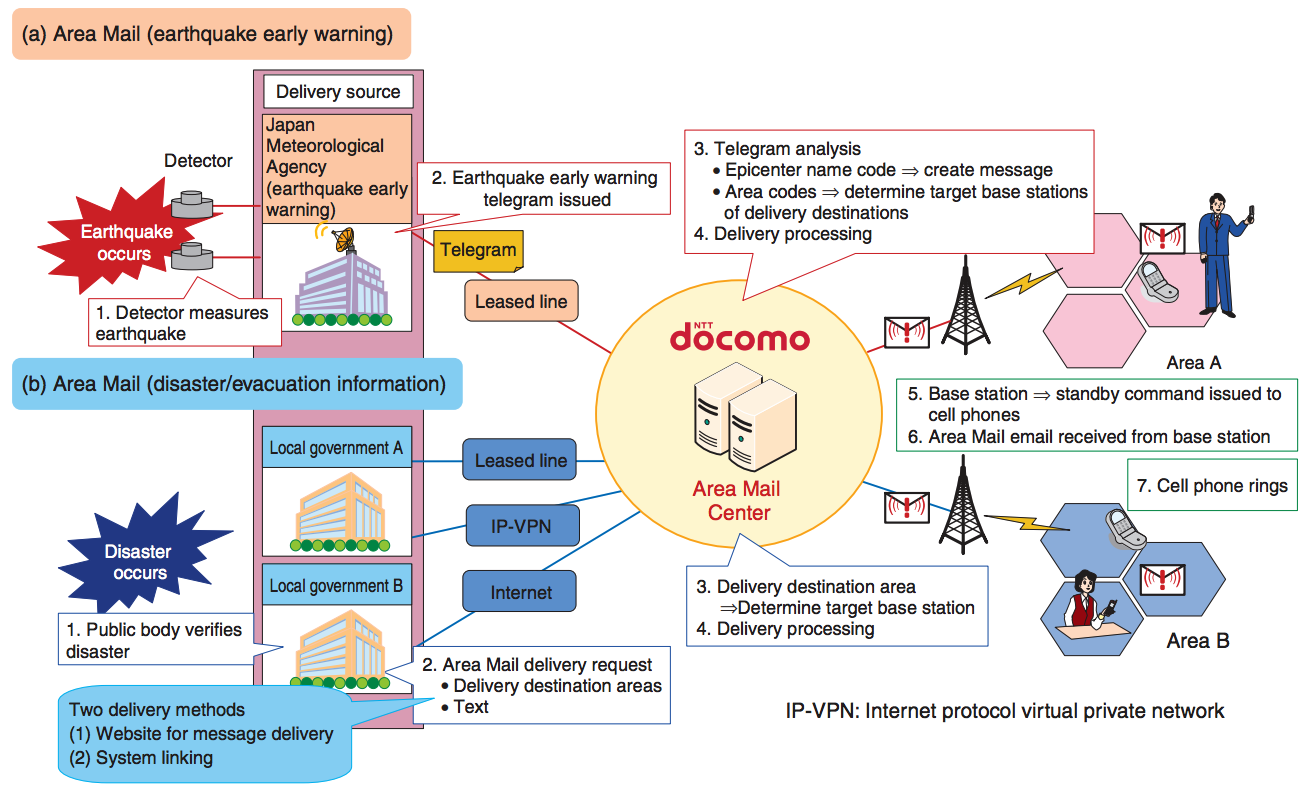
Quoting directly from the specification, which puts it well: *“The CBS service is analogous to the Teletex service offered on television, in that like Teletex, it permits a number of unacknowledged general messages to be broadcast to all receivers within a particular region. CBS messages are broadcast to defined geographical areas known as cell broadcast areas”*. You can think of it as a location-based service for cellular networks. From a carrier perspective there are multiple advantages in using such a system, including a lack of solid alternatives back at that time (SMS is the most obvious one) and even more importantly, a message system that is not impacted by network congestion due to regular subscriber activity. Let’s say a warning message shall be delivered during an otherwise busy event such as new year. Most of us probably experienced issues with the quality of services on that date. This is just an extreme example, but clearly there would be reliability issues when using SMS as a carrier. This is where CBS fills a gap.

Due to radio resource limitations, a CBS message can contain up to 82 bytes (3GPP and ETSI specs often refer to this as octets) of payload. Assuming a standard GSM alphabet and 7-bit encoding such as used in SMS, this translates to 93 characters. That’s not a whole lot, but when switching between a set of messages that are repeated in cycles, this allows already for a reasonable amount of information. Phones can handle these messages analogous to short messages, which are also transmitted on an SDCCH, but carry additional information (such as the sender number). The interested reader can find an up-to-date specification of the Cell Broadcast Service [here](http://www.3gpp.org/ftp/Specs/html-info/23041.htm). At the time of introducing CBS, other than providing the capability, there was no particular settled use case for the system.

**Disaster and Evacuation Warning Systems**

CBS already carries meaningful complexity from a security perspective by itself. However, historically there was a need for other systems, which specify concrete use cases on top of this broadcast capability, including further structures contained within messages. Over time various countries also developed a need for specific disaster warning systems that are tailored towards concrete threats within their country. These are usually systems deployed outside of the cellular network and make use of several delivery mechanisms such as TV, radio, but also cellular networks to reach as many citizens of the broader population as possible. As you can imagine, requirements for such a system differ heavily across different countries. Japan for example is historically concerned with large-scale earthquakes while other parts of the world may be more concerned about missile alerts and other threats. In fact, Japan built its own system early on.

In 2005, after events such as the [Chūetsu earthquakes](https://en.wikipedia.org/wiki/2004_Ch%C5%ABetsu_earthquake), Japan created a working group to work on an early warning system for earthquakes that was called [Area Mail](https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr200812sf2.html). The system was commercially deployed in 2007. Following is a picture that shows its general architecture.



**Figure 2: Area Mail - Copyright (C) Nippon Telegraph and Telephone Corporation**

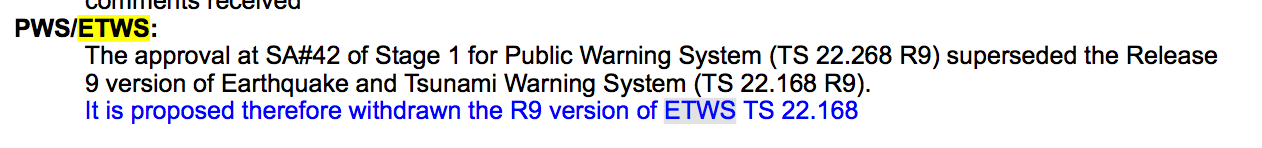
About the same time in 2007-2008, the US started developing a system called [Commercial Mobile Alert System](https://www.fcc.gov/general/commercial-mobile-telephone-alerts-cmas) (CMAS) in consequence of the [Warning, Alert, and Response Network Act](https://transition.fcc.gov/pshs/docs/emergency-information/cmas-warn-act.pdf) that passed Congress in 2006. This system was also based on CBS, but uses different message details.

In parallel, other systems, including systems based on SMS, were developed by a number of countries. Air Mail later became the Earthquake and Tsunami Warning System (ETWS) as support for Tsunamis was added to the system. The European Union developed the [EU-Alert](https://ec.europa.eu/programmes/horizon2020/en/news/emergency-alert-system-europe), Korea developed the so-called Korean Public Alert System (KPAS), Chile has the [ONEMI ‘LAT-Alert’](http://www.onemi.cl/tipos-de-alertas/), Israel has developed an alert [system](http://www.oref.org.il/Sip_Storage/FILES/4/1914.pdf) based on CBS, and even smaller countries, such as the Netherland, have their own warning system ([NL-Alert](https://crisis.nl/nl-alert)).

As the world is in crisis mode, more and more requirements were formalized and we will see these systems being continued to be developed over the years. Some of them using CBS, some relying on SMS.

**The Public Warning System (PWS)**

When first looking into warning systems, the sheer amount of different specifications and their different release versions was quite confusing. For example, while [3GPP TS 22.168 Release 8](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=619) specifies how ETWS [Stage 1](http://www.3gpp.org/contact/3gpp-faqs#S13) is supposed to work, Stage 2 and Stage 3 were missing. One could assume that a specific release version of 3GPP documents is consistent at least. However, this is not true. When doing cellular research, it is important to realize that information may be incomplete or even wrong. These documents are for the most part bleeding edge and subject to constant changes. In fact, Stage 2 ([3GPP TS 23.168](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=800)) is withdrawn. In order to understand what happened here, we can make use of public meeting reports of 3GPP. Particularly the [Draft report of TSG SA Meeting #43](https://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=397920) contains more information here.



**Figure 3: Excerpt from 3GPP SA Meeting #43 Report**

So what is this Public Warning System (PWS) that is referenced here?

Apparently, after the Indian Ocean tsunami in 2004 and hurricane Katrina in 2005, [ETSI](http://www.etsi.org/) started working on the definition of a unified Public Warning System (PWS) that should provide message security and authentication as well as perform even under network congestion situations. The GSMA has published a nice [write-up](https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2013/01/Mobile-Network-Public-Warning-Systems-and-the-Rise-of-Cell-Broadcast.pdf) about the historic context here. The requirements for such a system resulted in a [Study for requirements for a Public Warning System (PWS) service](http://www.etsi.org/deliver/etsi_tr/122900_122999/122968/09.00.00_60/tr_122968v090000p.pdf) (3GPP TS 22.968). PWS these days has become the basis of warning systems in cellular communication networks. It does not replace existing alert systems or require the use a specific alert system, but rather generalizes the concepts and adds additional requirements on how these systems are used in practice. The result can be found in the [Public Warning System (PWS) requirements](http://www.qtc.jp/3GPP/Specs/22268-b40.pdf) (3GPP TS 22.268), which explicitly integrates requirements for other systems such as ETWS and CMAS.

Most systems these days utilize cell broadcast to carry warning information and nation-state alert systems are mostly covered by PWS. Still, with crises all over the globe, exploration and specification work around warning systems is an ongoing topic as can be seen for example in recent [FCC documents](https://transition.fcc.gov/bureaus/pshs/eas/Earthquake_Alert_WhitePaper-120216.pdf). Moreover, not every entity that wishes to issue a warning has access to carrier networks. An example for this is a university campus alert system. Henceforth, warning systems carry the legacy of SMS-based third party alert systems. The latter come with their own set of [challenges](https://dl.acm.org/citation.cfm?id=2197321), but are less of an interest to us as these usually do not have support in cellular baseband implementations.

**PWS from an Attacker’s POV**

PWS is an interesting beast from an attacker’s perspective. Historic events and the inherent complexity of radio systems led to the creation of a multitude of complex requirements and specifications, which provide a rich attack surface. Furthermore, complexity not only lies within one part of a mobile device. Instead, as with SMS, a part of the feature set is handled directly within the cellular baseband, while a fair share is handled on the application processor side, i.e. iOS in our case, because eventually information needs to be displayed.

The cellular baseband is mostly responsible for collecting messages or chunks of messages on different channels and passing them on to the application processor. The application processor would then issue a visible warning to the user such as the one seen at the beginning of this article. For this research we were only interested in the cellular baseband side and leave code paths on the application processor side for future exploration.

With regard to attacking a cellular baseband, PWS brings the following high-level [requirements](http://www.qtc.jp/3GPP/Specs/22268-b40.pdf)/features:

1. Broadcast information to multiple users and repetition of messages
2. Support for multiple concurrent broadcasts
3. PWS shall support multiple language encodings
4. Receiving notifications shall be possible across different RATS and in different situations (e.g. during an active call)

1) implies that there has to be the concept of sequences to handle repeating messages or detect messages that were already seen by a device. 2) implies that there has to be the concept of contexts to some extent to differ between multiple streams of messages for different purposes received in parallel. 3) potentially raises the complexity of code that has to deal with parsing messages. This is mostly relevant on the application processor side, which would ultimately display a warning pop-up of some sort.

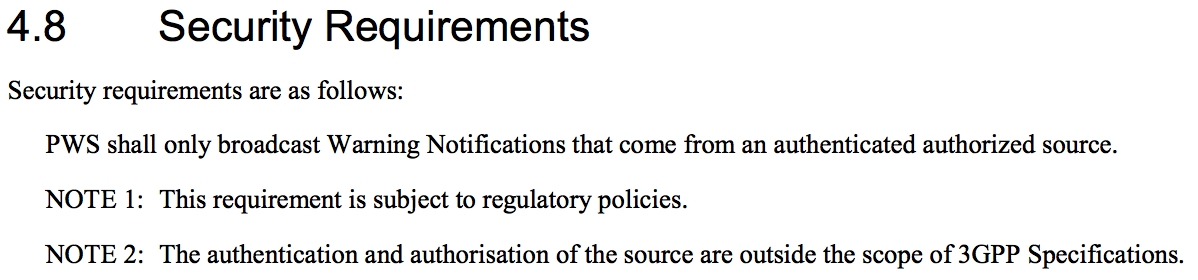
4) is the most interesting one, because it means that a device has to support message processing almost independently of what it is doing when an alert is issued, i.e. reception of messages needs to be supported in pretty much any state of the device. This leads to the involved protocols becoming more complex. The 3GPP [work item](http://www.3gpp.org/DynaReport/WiCr--370051.htm) for ETWS gives an idea of how many 3GPP specifications required changes in order to support ETWS alone. Furthermore, as the system is supposed to support different radio technologies, it also needs to support different types of encodings. Where TLV-like information elements and [CSN.1](https://en.wikipedia.org/wiki/CSN.1) encodings drive GSM and GPRS, ASN.1 encoded structures are used in LTE.

Additionally, a number of implicit requirements that have an impact on code complexity. For example the nature of radio messages play an important role. Radio messages - especially in GSM - are of very limited size. Especially in combination with 2) that means that there has to be the concept of message fragmentation and reassembly. Even more so considering that not only has alert text to fit into the limited space, but also control information for implementing 1-4.

**PWS Security**

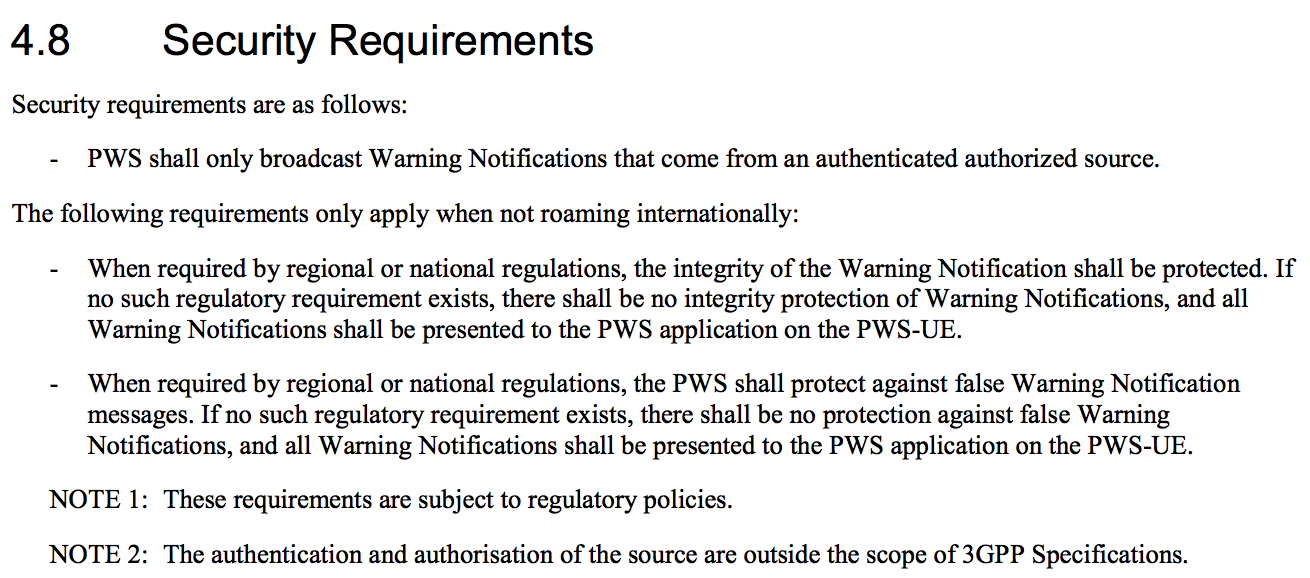
One may wonder whether carriers are able to secure the transmission of PWS-related alerts, e.g. through MACs. During the events of the false missile alert several researchers highlighted that such transmissions should be authenticated in the first place anyway. While this would not necessarily prevent false alerts, it certainly would be great from a security perspective. However, these wishes also serve as an interesting example of why security can be tough for the cellular standardization bodies.

In 3GPP TS 22.268 [Release 11](http://www.etsi.org/deliver/etsi_ts/122200_122299/122268/11.05.00_60/ts_122268v110500p.pdf) from 2013, we can find the following.



**Figure 4: PWS Security in Release 11**

As we can see in Figure 4, while it was a goal of the 3GPP to have alert systems with message authenticity, eventually it became an optional feature. This is a classic case of what happened historically at various places within radio technology specifications. Optional encryption in GSM or the introduction of a NULL integrity algorithm (EIA0) in LTE are other examples for that. Looking at the next release of the same document provides a rare insight into why that is.



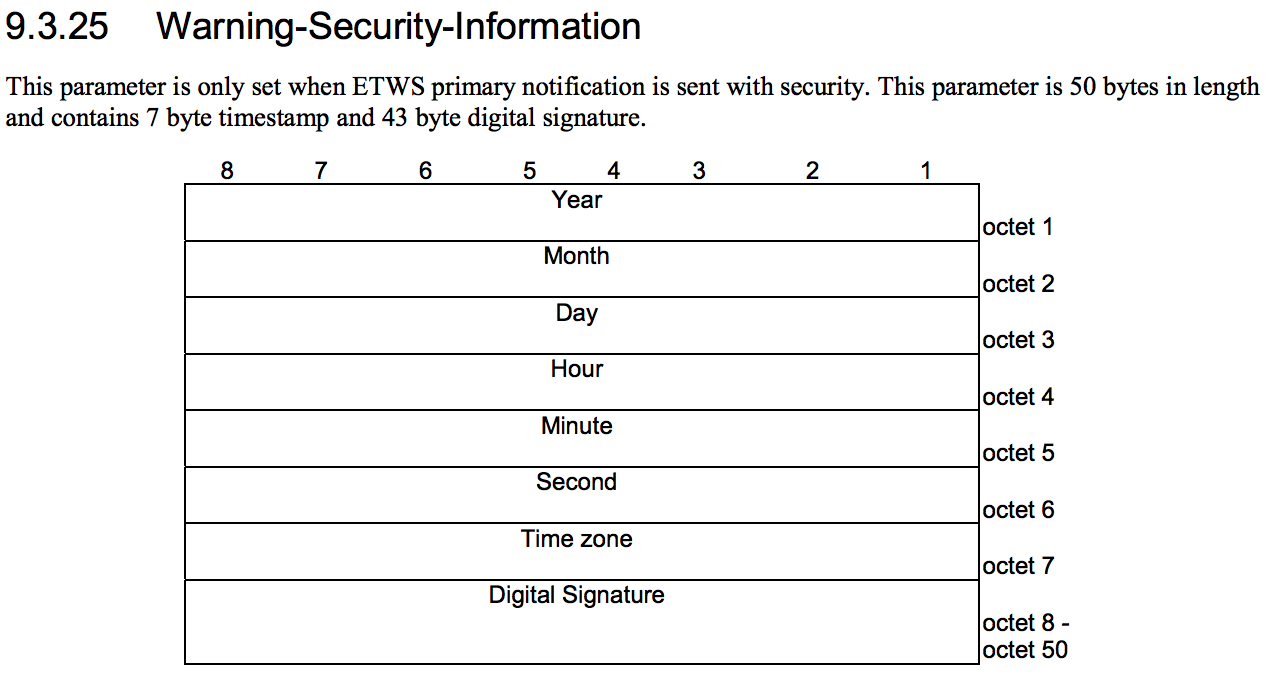
**Figure 5: PWS Security in Release 12**

The changes explain that the problems related to security all come from essentially two aspects that are a recurring pattern within the world of cellular security:

1. Countries that intentionally want weak or no security
2. Roaming subscribers

The second aspect is technologically the more challenging aspect, because it is essentially not clear how a device of a roaming subscriber would authenticate messages from another carrier for these specific cases. At the same time, it is certainly the intention of carriers to have a usable system for such subscribers. An extreme argument here would be that people should not have to die because of an unauthenticated message.

Does this mean that none of these systems supports security? Certainly not. ETWS for example provides digital signatures of the content as shown in Figure 6.



**Figure 6: ETWS Warning Security**

[3GPP TR 33.969](http://www.etsi.org/deliver/etsi_tr/133900_133999/133969/13.00.00_60/tr_133969v130000p.pdf) then shows that standardization bodies were considering the use of 128-ECDSA or 128-DSA signatures. Of course this does not include a solution for the problem of key distribution. The document then continues to discuss problems with the suggested feature set, which touch on the implicit requirement mentioned above.

*If the security solution is going to support ETWS Primary Notifications over GERAN then the total length of the signature and related security parameters cannot exceed 75 bytes.* *This limit rules out the possibility of including a certificate with the signed warning message, even when the certificate is stripped down to a bare minimum and only includes the subject public key and the issuer signature. However, so called implicit certificates can meet this length restriction at the expense of limiting the security level to 112 bits. Furthermore, the length limit also implies that RSA cannot be used as signature algorithm. Recall that the length of an RSA signature is equal to the length of the RSA key, which at the 128 bit security level is 3072⁄8=384 bytes long.*

***If ETWS Primary Notifications only need to be supported in UTRAN and E-UTRAN or not supported at all, then there is significantly more space available for the signature.”***

So because of size limitations, warning security is a problem in GERAN (GSM + GPRS/EDGE).

These are great examples for the balance that standardization bodies such as 3GPP and ETSI need to value. Whether these bodies are solving the right technological challenges remains an open question of course.

In any case, considering these aspects, it as a surprise to any reader that in practice such warning systems do not come with warning security enabled. We would be surprised if these messages are authenticated anywhere in the field. As a result, these alert systems are entirely open for an attacker with a rogue base station and physical proximity to vulnerable devices. Furthermore, the actual payload of the alert message is parsed on the application processor side, thus also providing a potential attack route that may not require jumping from one core to another.

With all that said, let’s actually look at an implementation vulnerability using the Intel baseband as used in iPhones as an example.

**Intel XMM Modem - ETWS Primary Notification Reassembly Overflow**

In summary, a combination of an integer underflow, a logic problem, and a lack of bounds checking leads to a memory corruption vulnerability when processing a stream of ETWS primary notifications contained in a sequence of paging messages.

To fully understand the vulnerability, we still need to add a bit of ETWS knowledge on top of what we have already covered around warning systems and PWS. We will try to keep it as short as possible and focus only on the aspects that matter for the vulnerability.

**Earthquake and Tsunami Warning System (ETWS) Background**

ETWS was introduced by Japan as a means to inform the public about earthquake and tsunami emergency situations. Unlike other public warning systems, ETWS differs between information that needs to be immediately available to citizens and information that is related to an initial warning, but not as urgent. Because of this, [3GPP TS 22.168](https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=619) introduces so-called **Primary Notifications** and **Secondary Notifications**. The purpose of the primary notification is to notify users within seconds of an imminent occurrence of e.g. an earthquake. The secondary notification carries supplementary information such as the seismic intensity or other helpful information.

In 2G and 3G, ETWS utilizes the Cell Broadcast Service (CBS) outlined before. This also explains the split between primary and secondary notification, because phones usually do not monitor cell broadcast channels unless they are in idle mode (see [3GPP TS 23.041](http://www.etsi.org/deliver/etsi_ts/123000_123099/123041/11.04.00_60/ts_123041v110400p.pdf)). CBS wasn’t designed specifically for providing a system for immediate alerts. This means that a phone that is currently in a call, receiving a text message, exchanging data packets etc. would not be able to monitor the CBCH. As a result, if cell broadcast is exclusively used as a warning transport, devices may miss alerts. To work around this problem in general, cell broadcast messages are usually repeated in fixed intervals. However, a timely notification such as wanted for ETWS primary notifications needs additional support. In LTE this mechanism is slightly different and as described in [3GPP TS 36.331](http://www.etsi.org/deliver/etsi_ts/136300_136399/136331/10.01.00_60/ts_136331v100100p.pdf) the *System Information Block (SIB) 10* is used to carry secondary notifications.

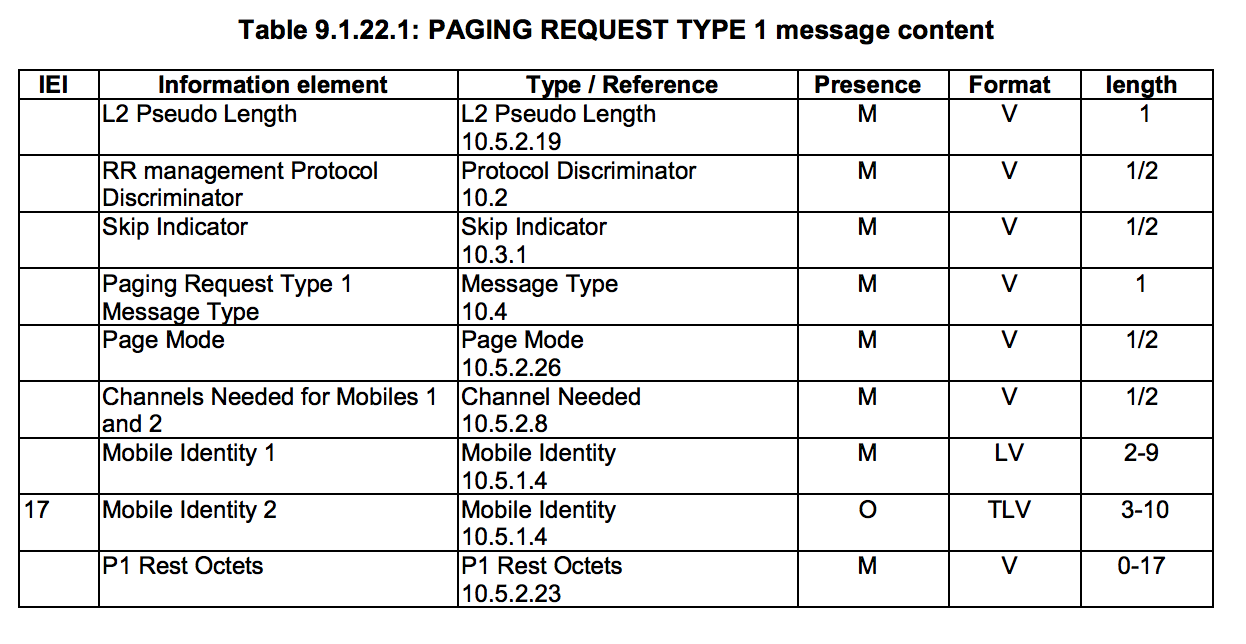
As the primary notification has to reach devices in different states, including connected states, the so-called *Paging* mechanism is used for ETWS. Paging is best known for providing the carrier with a mechanism of notifying a mobile device of an incoming service. This includes signaling mobile devices about incoming calls or short messages. The mechanism exists universally in all RATs. Paging messages can carry additional information however, including ETWS primary notifications. As paging messages are also received by mobile devices in connected states, it provides a mechanism to deliver primary notifications in both idle and non-idle states. After receiving a primary notification, devices continue to monitor the cell broadcast channels for secondary notifications.

Next, it is important to understand that there is not one specific over-the-air encoding of an ETWS primary notification. Instead, the exact encoding of messages differs between RATs. While LTE for example uses ASN.1 encoded information to construct paging messages, GSM and GPRS utilize CSN.1. Furthermore, depending on the exact message carrying the ETWS primary notification, encoded elements and size can be different. For example GPRS also offers a Packet Application Information message (see [3GPP TS 44.060](http://www.etsi.org/deliver/etsi_ts/144000_144099/144060/09.03.00_60/ts_144060v090300p.pdf)), which is missing ETWS fields that are present in GPRS paging. From an implementation perspective this means that there are multiple code paths involved in handling encoded ETWS messages from different sources.

So let’s have a look at how an ETWS primary notification looks like using GSM as an example.

**ETWS Primary Notification (GSM)**

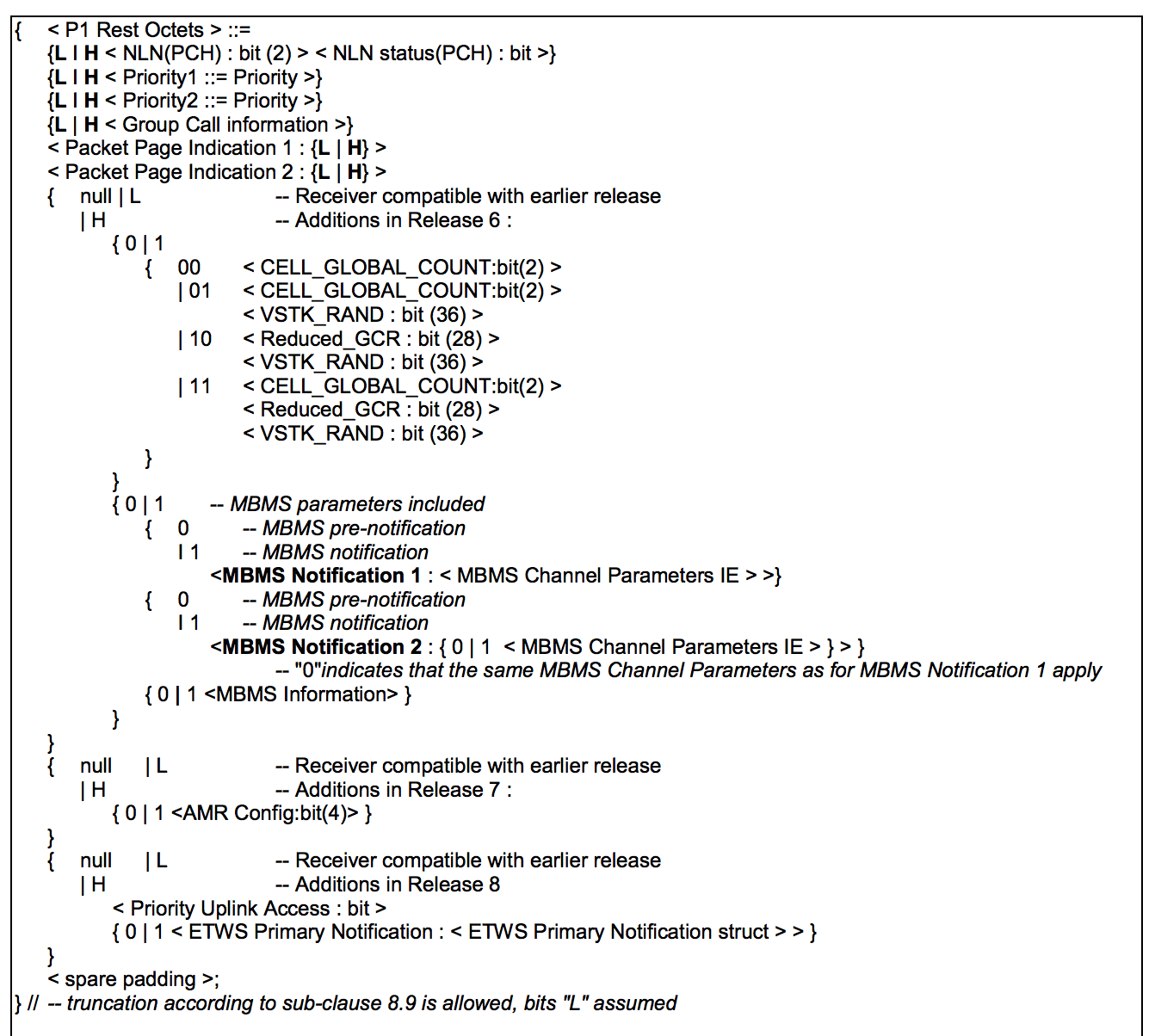
GSM paging messages are covered in the *Radio Resource Control (RRC) protocol* ([3GPP TS 44.018](http://www.etsi.org/deliver/etsi_ts/144000_144099/144018/10.03.00_60/ts_144018v100300p.pdf)). We are interested in *Paging request type 1*, which includes the ETWS primary notification data.



**Figure 7: Paging request type 1**

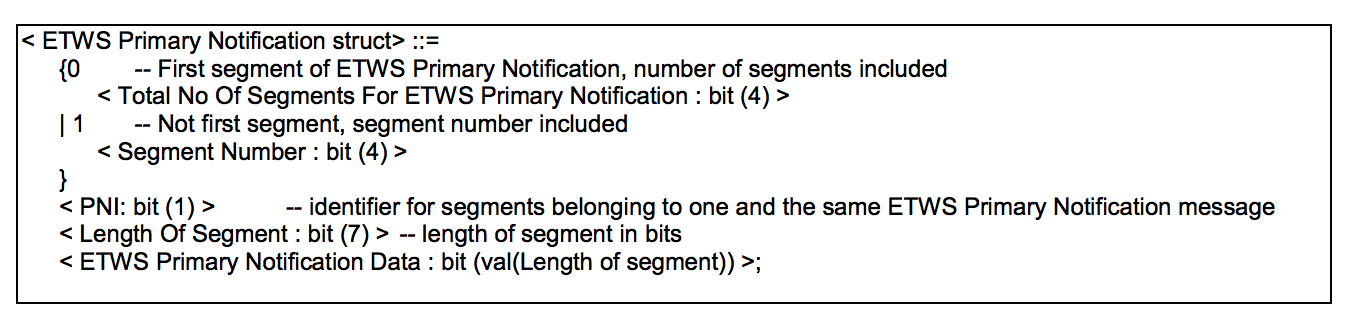
Figure 7 shows how such a paging request message is encoded. Except *Mobile Identity 2*, all fields are mandatory and either encoded as a plain value (byte sequence with fixed size), length value, or type length value (TLV). The length values specify the number of bytes/octets consumed by the element. The ETWS information is part of the so-called *P1 Rest Octets*, which has a variable length even though being a value field, which are usually of a fixed length. This is because these rest octets usually contain bits to pad a message. In most cases it consists only of a sequence of 0x2b and is rarely used. For completeness, if you recall past [research](https://events.ccc.de/congress/2011/Fahrplan/attachments/1994_111217.SRLabs-28C3-Defending_mobile_phones.pdf) on sniffing in GSM, this exact pattern (albeit in other message types) is what reduced to the attack complexity of breaking encryption due to the presence of a known plaintext. Either way, for a paging message these rest octets can also contain payload and this is what is used to implement ETWS primary notifications.

The P1 Rest Octets itself are encoded the Concrete Syntax Notation One ([CSN.1](https://en.wikipedia.org/wiki/CSN.1)). Also, just as a side-note, it is interesting to see that the entire cellular industry builds upon a standard, which was developed by one person, distributed as a book and hosted on a personal website, which by now is only available in the Internet [archive](https://web.archive.org/web/20060208070637/http:/perso.wanadoo.fr/cell.sys/csn1.htm). Other resources such as [csn1.info](http://csn1.info/) only provide an interpretation, not a complete official specification. It will be enough to understand the message though. Essentially CSN.1 provides a way to structure information as a stream of bits, without type information.



**Figure 7: Paging request type 1**

This CSN.1 specification provides a number of different options to be transferred within the P1 Rest Octets of which most are entirely optional, including the ETWS Primary Notification : < ETWS Primary Notification struct >. For the sake of understanding the notation and not diving too much into CSN.1, you can think of { 0 | 1 < ETWS Primary Notification : < ETWS Primary Notification struct > > } as either there is a 0 bit or if there follows a 1 bit, it will be followed by ETWS information.



**Figure 8: CSN.1 Encoding of ETWS Primary Notification in P1 Rest Octets in Paging Type 1 Message**

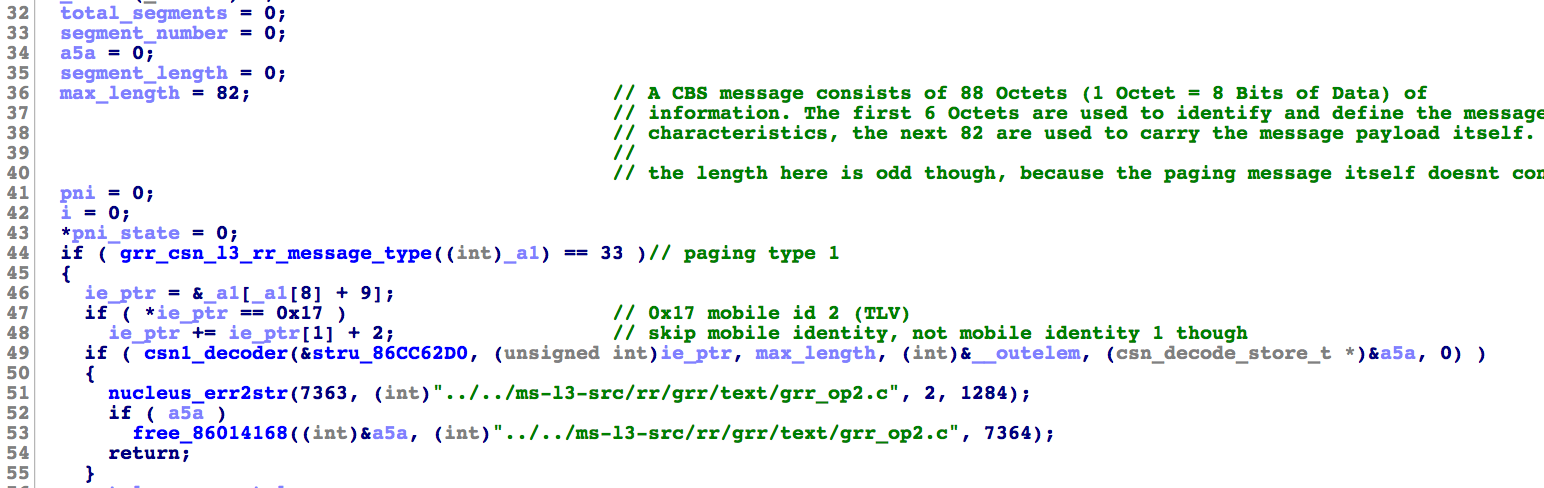
Figure 8 shows how the ETWS primary notification is encoded within the P1 Rest Octets. Remember, all of this information has to be contained within the maximum of 17 bytes that the P1 Rest Octets offer. Because that means that there is very little space, the ETWS primary notification in GSM absolutely has to provide support for segmentation. This is different for example in the aforementioned Packet Application Information message, which does not include e.g. a segment number.

Based on the specification, an ETWS primary notification can present its payload in two shapes. It either contains a first segment or does not. If it is the **first segment**, it starts with a 0 bit and is followed by a 4 bit value denoting the **number of segments to receive**. If a segment is **not the first**, it starts with a 1 bit and is followed by a 4 bit **segment number** value. In both cases, a 1 bit PNI value follow (which is used to fulfill the requirement of providing support for multiple warning broadcasts, here 2), a 7 bit length value for the content to follow, and a variable number of bits that depend on the given length that contain payload.

Now keeping in mind that warning systems will repeat messages and due to the small size of messages, things can become complex. For example, the cellular baseband can not assume that a transmission always starts with the first segment. For now this is enough information to understand the basics of the vulnerability.

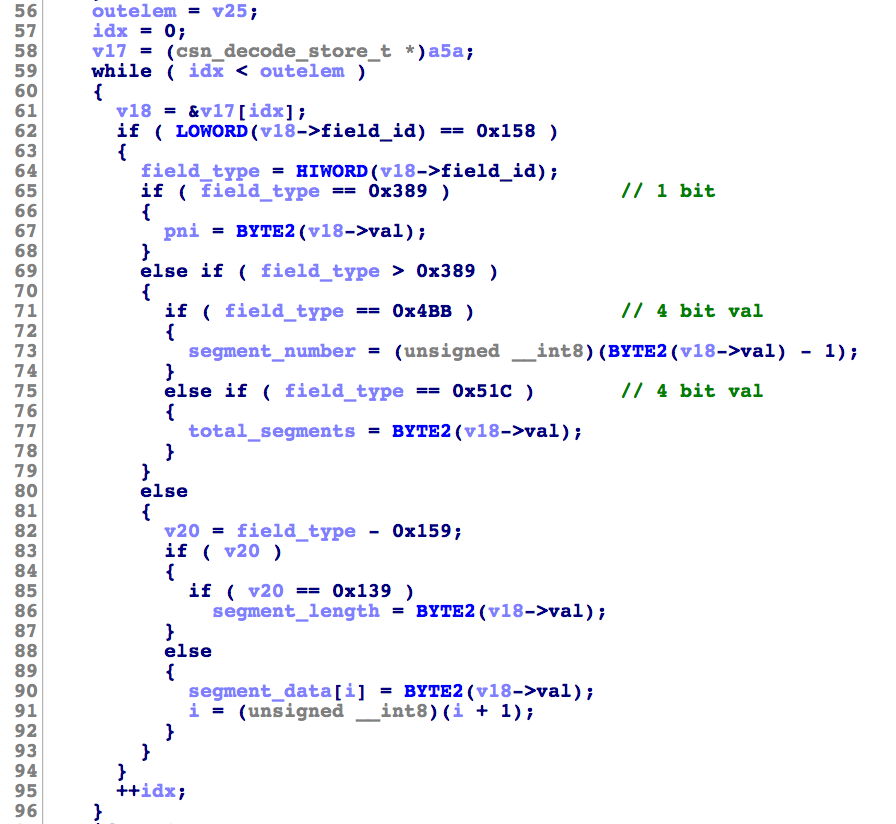
**XMM ETWS CSN.1 Handling**

With the basics of PWS, Paging, and ETWS, we will now walk through the respective code paths within the Intel modem as used on the iPhone. We start our analysis within a function that we call grr\_gsm\_rrc\_paging\_type1\_etws.



**Figure 9: Start of grr\_gsm\_rrc\_paging\_type1\_etws**

Figure 9 shows the start of the function, which receives a pointer to the plain over-the-air payload as the first argument (\_a1 here). grr\_csn\_l3\_rr\_message\_type makes sure that the message is indeed a radio resource message and is of type paging type 1. After this is done, the code skips other fields up to a potential Mobile Identity 2 field. The type value of that information element is 0x17. In case it’s present, the pointer is advanced again by the size of the element. Note that we did not dig into potential overreads here, because as an attacker we would have very little advantage here parsing bogus CSN.1 rest octet bit streams. It is important to note that segment\_number is initialized to zero; we’ll see shortly why. Based on our reverse engineering we know that csn1\_decoder receives a pointer to an array (stru\_86CC62D0) as the first argument, which is used to parse the CSN.1 elements into data structures. These structures tell us the size in bits of a particular parsed value. It also gives away which internal field type is used to store the value. The second function argument points to the over-the-air payload that is parsed. The fifth argument (a5a) will become a pointer to heap memory that is allocated for output within csn1\_decoder.

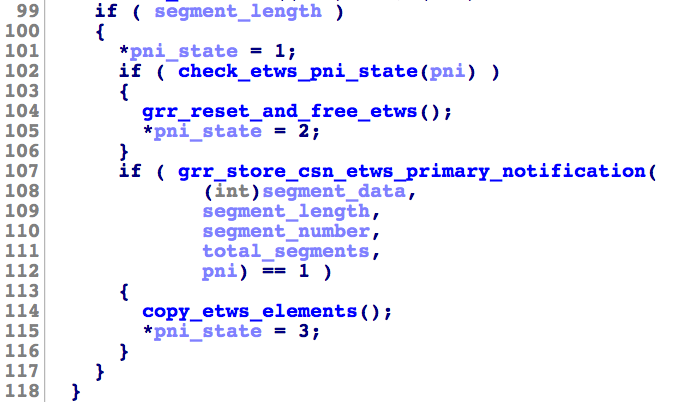


**Figure 10: Handling of Parsed CSN.1 Values in grr\_gsm\_rrc\_paging\_type1\_etws**

The csn1\_decoder call creates a storage structure for each parsed CSN.1 field that contains a type and a value. As shown in Figure 10 these structures are then evaluated in the code for handling paging messages. As far as we are aware there is no mapping between the internal type values and the specifications. We only know the purpose of the field types based on putting together our understanding the specifications, the bitsize contained in the CSN.1 arrays, and reverse engineering the underlying handler code. Based on which fields were present in the message, local variables are filled. segment\_data is handled differently as it contains a sequence of octets. We will not go into details of the CSN.1 implementation here, but it is worth noting that there is no overflow here and the implementation makes sure that there are as many 0x159 fields as indicated by the segment length value. For completeness, segment\_data provides space for 16 characters.

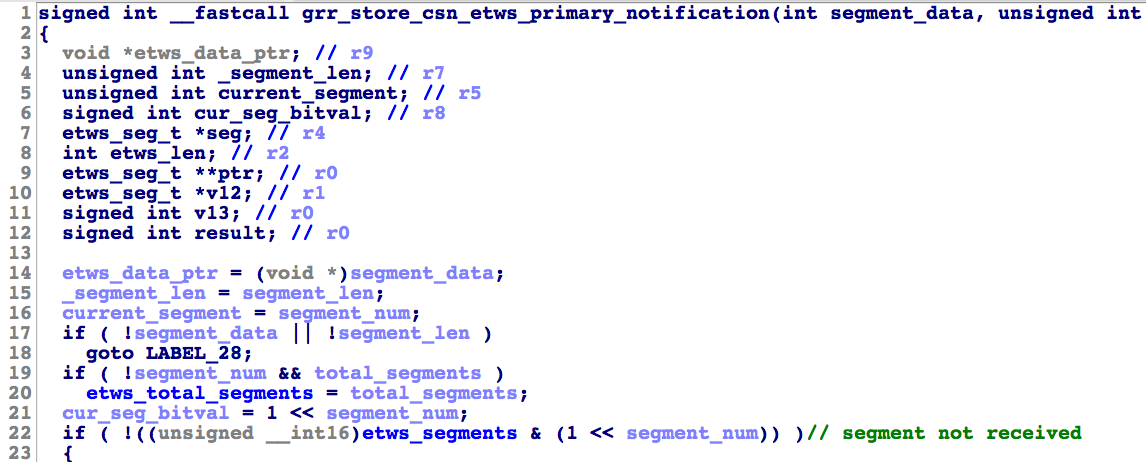
The first important aspect that our vulnerability leverages is that for an ETWS primary notification without segment number, a segment number of zero is assumed due to the initialization code.

The second important bit here is the assignment of segment\_number in line 73. Based on the specification, the first segment is handled differently from subsequent elements, which reflects in subtracting 1 from the segment number. We will see in the following function calls what the intention of that was. In any way, as segment\_number is an unsigned integer and the resulting value is cast to an unsigned int8, the code introduces an integer underflow with subsequently truncating the underflown value to a byte. As a result, segment\_number can become 0xff if a segment number of zero was encoded as part of the CSN.1 sequence. This should not happen normally as zero is implicitly assumed for the first segment.



**Figure 11: Main Logic Around Storage of ETWS Primary Notification in grr\_gsm\_rrc\_paging\_type1\_etws**

Figure 11 shows that the code next differs between PNI values to either free received ETWS data or call into grr\_store\_csn\_etws\_primary\_notification and based on its return value call into copy\_etws\_elements. This matches our understanding of the specification as the code has to handle two concurrent transmissions of alerts. The code treats segments with the same PNI bit as belonging to each other.



**Figure 12: grr\_store\_csn\_etws\_primary\_notification**

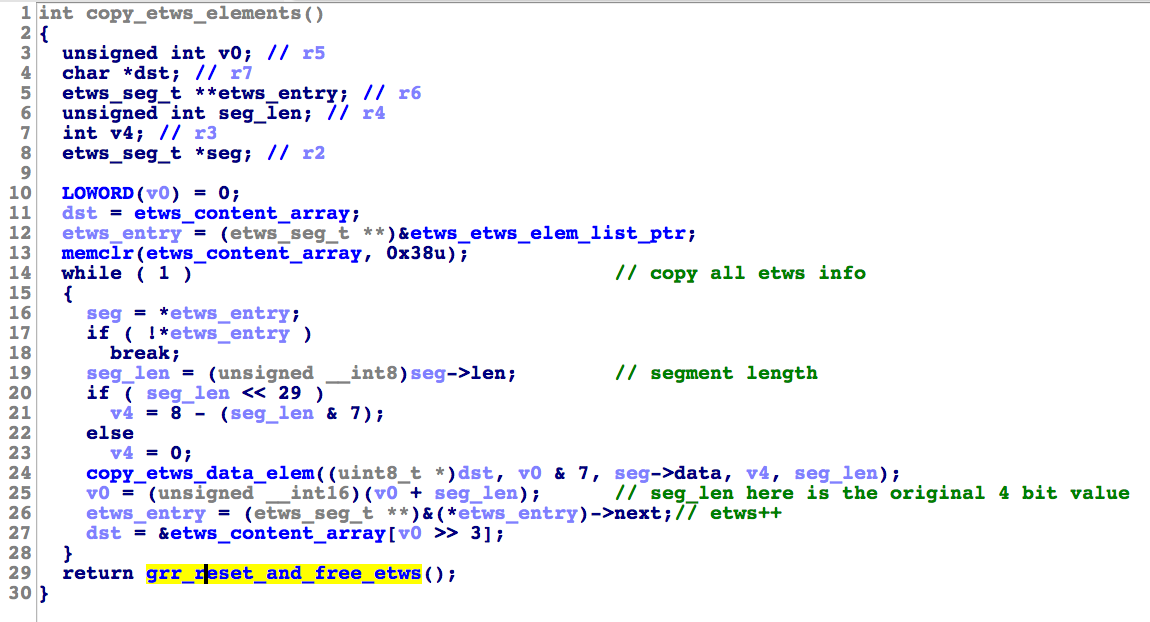
grr\_store\_csn\_etws\_primary\_notification next uses the segment number to create a bitmask for segments that were already received (line 21). Line 22 then checks if the bitmask already contains the segment and if not, continues to process it.



**Figure 13: grr\_store\_csn\_etws\_primary\_notification Logic for ETWS Primary Notification Segment Storage**

If the segment was not received, grr\_store\_csn\_etws\_primary\_notification continues to allocate an object on the heap, which will be used as a linked list element. Starting in line 35 the code traverses that linked list and adds the element to the list in line 43. To track which ETWS primary segments were received, the global bitmask etws\_segments is updated in line 48. Finally, once all segments were received according to the bitmask, the function returns 1 in line 57.

Now recalling the details from Figure 12, once grr\_store\_csn\_etws\_primary\_notification returns 1 and indicates that all segments were received, copy\_etws\_elements will be called.

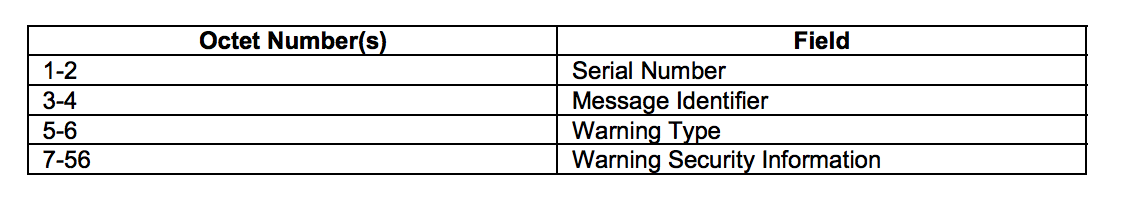


**Figure 14: copy\_etws\_elements**

Once all segments are present, copy\_etws\_elements goes through the list of ETWS segments another time (via etws\_etws\_elem\_list\_ptr). It then uses the length of the segment from the list (line 19) and uses copy\_etws\_data\_elem to copy the respective number of segment data bits to a pointer, which points into an etws\_content\_array, which lies within the data segment of the modem.

**Specification vs Code** The first aspect to notice in this function is that etws\_content\_array seems provide 0x38 bytes of content (based on the memclr call in line 13). This seems slightly arbitrary given that we have seen before that the segment data length is 7 bits and the segment number is a 4 bit value, i.e. 240 bytes in total. This seems really weird and would also introduce a buffer overflow. For a while we were puzzled about that particular number of bytes and now believe that it serves as a great example of where the 3GPP specifications fall short and show how it might be easy from a developers perspective to introduce such vulnerabilities.

First, what is this data buffer in the first place, is it already containing text? To our surprise, the answer can be found within the *Technical realization of Cell Broadcast Service (CBS)* [3GPP TS 23.041](http://www.etsi.org/deliver/etsi_ts/123000_123099/123041/11.04.00_60/ts_123041v110400p.pdf) specification (cause where else? ;).



**Figure 15: ETWS Primary Notification Message Content**

Figure 15 shows that the array size is all but arbitrary, but in fact provides the exact amount of bytes as required by the 3GPP specification. It would be plausible that an Intel engineer used the size from the specification when allocating the etws\_content\_array buffer, which matches exactly that size. Now consider a situation in which more than one engineer is tasked with implementing that feature. One engineer handles 23.041, while the other engineer handles 44.018, which contained the CSN.1 notation. In that case it is easy (or at least plausible) to introduce such a flaw, because who of the two should be responsible for adding a bounds check? Well ideally both of course, but reality often works different. Clearly this is an example of poor specification work by 3GPP.

The code is actually worse than that so this is not meant to pick on 3GPP, but clearly there is space for improvements here.

**Logic Flaw + Underflow + Overflow** Considering the integer underflow that we have described before and the bit mask logic, the code actually introduces a more critical memory corruption problem. Specifically, as visible in Figure 12 and 13, etws\_segments, which is the bitmask to track received segments, is only set once a segment was not received already. This logic is perfectly fine, however combined with the underflow resulting in the potential for segment\_num to be 0xff, this introduces a primitive that allows in theory to send an infinite number of ETWS paging message segments. Imagine a sequence such as the following one.

1st ETWS Segment (number: 0)

2nd ETWS Segment (number: 0)

3rd ETWS Segment (number: 0)

...

nth ETWS Segment (total segments 1)

Every time a segment with number 0 is received if ( !((unsigned \_\_int16)etws\_segments & (1 << segment\_num)) ) would be true. This is because 1 << 0xff is also zero. At the same time, the number of total segments to receive (total\_segments) would stay zero initialized until the first segment announcing the total number of segments was received (as per CSN.1 specification). Therefore, the function would also leave etws\_total\_segments at zero until the *nth* segment indicates a total of 1 segment, which would immediate cause grr\_store\_csn\_etws\_primary\_notification to return 1, irrespectively of the fact that a number of segments were received and added to the list before. Once that happens, copy\_etws\_data\_elem not only introduces the trivial buffer overflow that we described before, but effectively allows to write a lot further from the start of etws\_content\_array by sending further paging messages and abusing the logic of the code, the integer underflow, and the lack of bounds checking in either of the involved functions. The content of written memory is completely controlled by an attacker.

Recall that we mentioned that there are a number of different encodings of primary notifications in different RATs. There certainly is a lot of complexity hidden here and it is worth noting that the GPRS code path is handling *Packet Paging Messages* is equally vulnerable to these issues.

We believe that the combination of these issues allows the execution of arbitrary code within the Intel modem by an attacker running a rogue base station and physical proximity to a victim.

**Exploitation Hurdles**

Exploiting cellular radio stacks can often be tricky for multiple reasons. In our [past research](https://comsecuris.com/slides/recon2016-breaking_band.pdf) we have already outlined a few aspects on how to approach reverse engineering such a target in the first place. This particular vulnerability however also highlights interesting aspects related to the nature of cellular protocol stacks itself.

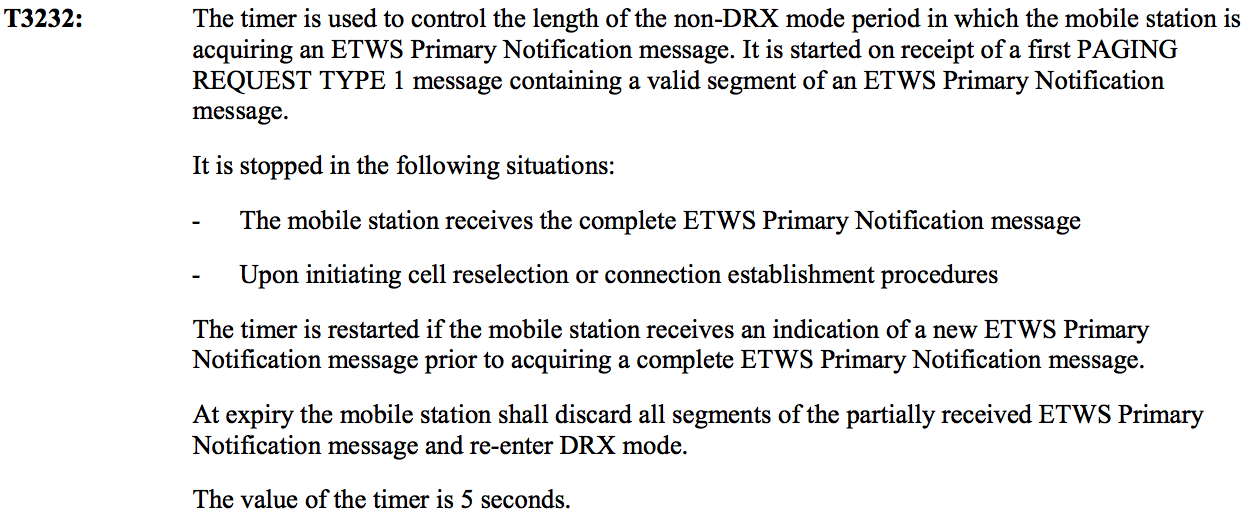
**Discontinuous Reception (DRX)** [3GPP TS 43.013](http://www.etsi.org/deliver/etsi_ts/143000_143099/143013/10.00.00_60/ts_143013v100000p.pdf) introduces the concept of DRX. DRX is essentially a way for mobile devices to save battery power, while idling within a network. As described before, the bulk of paging activity in non-crisis scenarios is used to notify devices of incoming services. Naturally, most paging request messages are destined to other subscribers. Therefore, when a device is idling, it makes sense to optimize channel monitoring in a way that allows to only read paging messages when it is likely that the message contains the mobile identity of your device. This is exactly what DRX implements both on the network side and the phone side. It is a mandatory feature of GSM. It essentially provides a way for the carrier to divide paging messages for its subscribers between certain transmission time slots, based on the mobile subscriber’s identity. Mobile devices then only have to monitor time slots relevant for their own identity when idling and as a result save battery power.

This aspect is important when exploiting this issue with a custom base station solution, because it means that precise control over the amount of memory that is corrupted requires a proper implementation of DRX. Otherwise, the segment carrying the first payload sequence might be missed by the target device.

**Spatial Constraints** We saw that the P1 Rest Octets provide up to 17 bytes of payload. A fair share of its content however is meant to encode other fields. If we craft a minimal message that is valid by the specification, we end up only having 13 bytes left for each segment that we transmit as part of an exploit. Considering this, even with proper bounds checks, the code would actually also provide more content than actually required by the specification in reality. Spatial constraints are frequently encountered when exploiting issues in GSM. As Ralf also outlined in his [research](https://comsecuris.com/slides/Weinmann-All-Your-Baseband-Are-Belong-To-Us-slides.pdf) before, the size of a Layer 3 frame in GSM is dictated by a byte field (N201) and can provide a maximum of 252 bytes.

**Timers** Cellular protocol stacks are driven by complex state machines and the specifications include tons of different timers for various purposes. Now recalling that the intention of ETWS was to deliver the primary notification as quickly as possible, the protocol specifications of course come with a timer: T3232. While experimenting with the vulnerability we were at first only aware of ETWS coming with time constraints compared to other warning systems, but only noticed during monitoring modem memory that some parsed fields get reset to zero even without changing the PNI bit value. At that point, we saw this happening roughly after a few seconds.

[3GPP TS 44.018](http://www.etsi.org/deliver/etsi_ts/144000_144099/144018/10.03.00_60/ts_144018v100300p.pdf) explains T3232.



**Figure 16: ETWS T3232 Timer**

Knowing that, we also later identified the code that implements this timer. Anyway, in conclusion this means that as an attacker, you have 5 seconds for exploiting this issue in practice. You may wonder how many ETWS segments you can realistically send in that time frame and how far you can write. As often in the space of cellular networks, the answer would be *“it depends on the network”*. The reason for this is that there are a number of different channel configurations for carrier networks, which have an influence on how many logical channels for a particular purpose, including the number of paging channels, is available. The details go way beyond the scope of this article, but the interested reader can learn more about this in [3GPP TS 05.02](http://www.etsi.org/deliver/etsi_gts/05/0502/05.00.00_60/gsmts_0502v050000p.pdf). During our experiments, we were able to write up to around 1400 bytes beyond the end of the array.

**Conclusions**

We hope that this post adds another useful bit of information to the research community within the complex and exciting world of cellular communication. Even though we are all ultimately affected by the security of cellular basebands and these systems are around for a long time, they still represent a rather unpopular area of (public) research. While most academic research still focuses on protocol level security, we hope that this article also showed that there is significant space for software security in this space. Likewise, cellular radio systems provide unique challenges in terms of true remote exploitation.

3GPP specifications can be tough to understand at times, especially when not all specifications belong to a particular feature are known. They are not meant to serve as a tutorial, but a reference for requirements to implementations. As we have shown, this can lead to subtle problems, inconsistencies, and even vulnerabilities.

The illustrative example of warning systems furthermore helps to understand that even in a world with proper mutual authentication and integrity protection of messages, there is still a significant amount of complexity in parts of the stack that have to be unauthenticated by design. Furthermore, this is yet another aspect illustrating why simply deprecating 2G is challenging, e.g. due to roaming subscribers. Looking at the ongoing standardization work, it is unclear whether specification work around seemingly legacy features can be considered as finished or suspended.

**Affected Devices**

We believe the vulnerability is present in Intel’s cellular baseband solution since at least XMM7262, which was [introduced](https://ark.intel.com/products/77507/Intel-XMM-7262) in Q3’2014. Newer versions (XMM7360 and XMM7460) of the platform have since been used in Apple iPhone devices since the introduction of an Intel-based iPhone 7. We confirmed the vulnerability on at least the following devices:

* iPhone X (iOS 11.2.6, XG748ES20S6TE10FLUBMAV2DEFA17450002416)
* iPhone 8 Plus (iOS 11.2.6, XG748ES20S6TE10FLUBMAV2DEFA17450002416)
* iPhone 8 (iOS 11.2.6, XG748ES20S6TE10FLUBMAV2DEFA17450002416)
* iPhone 7 Plus (iOS 11.2.6, XG736ES21S5E20FLMAV2DEV5017456202233)
* iPhone 7 (iOS 11.2.6, XG736ES21S5E20FLMAV2DEV5017456202233)

In the above listing, the XG-versions refer to an X-Gold version, which is the trademark name of Intel’s cellular baseband implementation. The entire modem solution that also includes transceivers etc is referred to as XMM however. Older idevices that use [XMM6180](https://www.theiphonewiki.com/wiki/XMM6180) are not affected to the best of our knowledge.

**Vulnerability Timeline**

* 2018-02-19: Reported buffer overflow, integer underflow, and logic issue to Intel and asks for fix timeline
* 2018-02-20: Intel acknowledges receipt and asks for timeline from Comsecuris; report forwarded to relevant product team
* 2018-02-20: Comsecuris expects a maximum fix time of 60 days
* 2018-02-22: Intel responds with preliminary analysis: basic overflow issue known;issues being worked on
* 2018-02-22: Intel asks to resubmit issues over HackerOne. Comsecuris asks for rationale.
* 2018-02-23: Comsecuris requests more details on known aspects as issue is unfixed on fielded devices;
* 2018-02-23: As issue is unfixed on up-to-date iPhone devices, Comsecuris reports issue to Apple
* 2018-02-23: Comsecuris indicates intention to release full details with iOS update and asks for Intel’s timeline
* 2018-02-27: Intel clarifies awareness of basic overflow since Q4’2017; complete impact wrt underflow/logic issue apparently new
* 2018-02-27: Intel indicates that 60 day window is likely too short (despite being aware of the problem area since 2017)
* 2018-02-28: Apple acknowledges receipt of report; investigating issue
* 2018-03-02: Comsecuris declines HackerOne resubmission and requests more details from Intel on challenges and relevant industry partners
* 2018-03-07: Apple confirms intent to address issue in upcoming 11.3 release
* 2018-03-07: Intel requests time to address this issue until June 19th (120 days) without giving further details on ecosystem/customer situation
* 2018-03-07: Comsecuris explains this to be unreasonable and asks for further concrete details on ecosystem and challenges
* 2018-03-23: Apple assigns [CVE-2018-4148](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-4148)
* 2018-03-27: Apple [releases](https://support.apple.com/en-gb/HT208693) iOS 11.3
* 2018-03-28: Intel asks Comsecuris to sign an NDA for sharing customer advisory
* 2018-04-03: Comsecuris declines request and shares heads-up about upcoming article
* 2018-04-04: Intel asks for holding off disclosure until supposedly previously agreed date of April 17th
* 2018-04-04: Comsecuris releases article detailing vulnerabilities

As you can see, we have ultimately decided to only wait with the release of this information until Apple has rolled out iOS 11.3 (and enjoy Easter holidays). We did this for multiple reasons. First, we believe that requiring 120 days for addressing critical security issues is not acceptable, especially if there has been an indication of being aware of at least a subset of the issues for a longer time. In fact we did not expect something like this at all since the devices at hand were all unfixed.

At the same time, we think that a player such as Intel needs to be in a position to respond to such issues at a much faster pace, especially when its customers are able to. Since we do not believe that exposing users in the field to risk fulfills a greater good, we had to balance what we think is reasonable and protecting fielded devices. Comsecuris attempted to determine Intel’s industry partners on the relevant product lines and could not find major players besides Apple in the handset business. There are older Asus devices that receive no updates anymore anyway. Likewise, there are PCIe/M2 cards from Sierra Wireless and Fibocom. The former however seems [dead](https://zukota.com/sierra-wireless-em7445-a-dead-born-king/) (and superseded) while the latter does not support GSM. Also, the issue is publicly addressed by at least one customer so that further coordinating here is questionable. As a result, we believe that we rather uphold our standards on vulnerability disclosure than to follow a timeline that we think is not reasonable and also based on incomplete information.

We would like to thank Apple for the quick turnaround and professional handling of the issues!