

MEDICAL IOT FOR DIABETES AND CYBERCRIME

Axelle Apvrille & Aamir Lakhani
Fortinet, France & USA

{aapvrille, alakhani}@fortinet.com

ABSTRACT

This paper evaluates the threats diabetic patients face when they use smart glucose monitoring devices. We study those threats from three different angles:

1. The security and privacy of a smart glucose monitoring device. We have selected and analysed one of the most widely used systems. Contrary to the poor security levels we unfortunately often see in IoT devices, this device has been designed with care. It is not perfect, though (nothing is), and we discuss two issues.
2. Malware. Compromising a smart glucose monitoring device is only one of the ways in which such a system can be attacked – and perhaps not the most effective. Indeed, patients rely on their smartphones to read their glucose levels, and this only remains safe as long as the smartphone is not compromised. We uncover several diabetic-related pieces of malware. While they are scams and do not directly affect the health of victims, they do have side-effects that can impact the victim's ability to deal with their diabetes.
3. The Darknet. Our research reveals that cybercriminals have a strong interest in the medical sector. We have collected evidence of trading for many different treatments specifically for diabetes.

INTRODUCTION

The medical sector is certainly one of those domains which particularly deserves our attention. Numerous hospitals have been targeted by ransomware [1–4] and the trend has only been increasing with attacks on IoT devices [5]. For example, [6] relates the case of an infected spectrometer which was sending outband spam email and mining cryptocurrencies. These attacks are not as sensational as the assassination of the Vice President of the USA in the TV series *Homeland*, but they are *real*, and so is their impact (financial loss, privacy leaks, etc.).

We decided to study a practical case: what threats do diabetic patients using a smart device face? Diabetes, more precisely *diabetes mellitus*, is a 'chronic disease caused by inherited and/or acquired deficiency in production of insulin by the pancreas, or by the ineffectiveness of the insulin produced. Such a deficiency results in increased concentrations of glucose in the blood, which in turn damage many of the body's systems, in particular the blood vessels and nerves.' [7]. Diabetes is an interesting research topic for several reasons: (1) it is a life-threatening condition, (2) it requires daily attention, and (3) it affects up to 9% of adults worldwide.

The first part of our study concerns the security and privacy of an IoT device, a glucose sensor, which is popular among diabetic patients and commercialized around the world. Then we discuss diabetic-related malware and cybercrime.

SECURITY AND PRIVACY OF A FLASH GLUCOSE MONITORING SYSTEM

Diabetic patients have to measure their blood glucose levels regularly (e.g. four to six times per day) and decide whether to inject insulin or not, depending on the level. Typically, they prick their fingertip and measure blood glucose using a blood glucose meter. This repeated routine is tedious, and as a consequence, systems that *automatically* measure blood glucose levels are seen by many patients as a great relief. In such automated systems, a sensor is applied to the patient's skin, which measures the amount of glucose in interstitial tissue and transmits the data to a blood glucose reader. Note that testing glucose in interstitial tissue is not quite the same as testing directly in blood but it usually gives acceptable results [8]. These automated systems fall into two categories: Continuous Glucose Monitoring (CGM) systems and Flash Glucose Monitoring (FGM) systems. The main difference is that CGMs measure the glucose level continuously and require calibration, while FGMs only measure a few times per hour.

The FGM system we have investigated consists of two parts:

1. A glucose sensor. The sensor comes in two parts: (1) the actual enzyme sensor with a soft needle, and (2) the electronics.
2. A smartphone with a dedicated mobile application supporting NFC to communicate with the glucose sensor. As for the mobile application, there is choice between the official app and several others developed by independent and/or open-source communities: *Glimp*, *xDrip*, *OpenLibre*, *Liapp*, *Glycemia*, etc.

Sensor lifecycle

The glucose sensor is assembled then applied to the patient's skin. It must then be activated to become operational. This can be done via an NFC scan of the sensor with the official app or the *Glimp S* app. The sensor enters a warm-up period (one hour in Europe) before finally being ready to use. Note that the warm-up period exists for medical reasons, because when you apply the sensor the soft needle causes some (usually minor) local tissue disruption that the body heals with an inflammatory reaction that consumes glucose and hence alters the glucose readings [8]. The warm-up period allows time for the glucose signal to stabilize. The length of the warm-up period varies between countries depending on how strict the regulations are in the different countries [9].

The sensor expires after 14 days and must be replaced by a new one. This point will be detailed later. The glucose sensor is sold for approximately 60 euros.

As summarized in Figure 1, from a security and privacy angle, threats to this system are located:

1. In the sensor itself (e.g. hardware tampering)
2. In the communication between the sensor and the smartphone (NFC hacks)
3. On the smartphone: mobile app or compromise of the phone itself.

Hardware

While hardware attacks are theoretically possible (e.g. tampering with the sensor to produce different glucose readings), they seem both difficult to conduct and of limited interest in practice. Therefore, we will quickly run through this part, mainly because it helps understand the sensor's architecture.

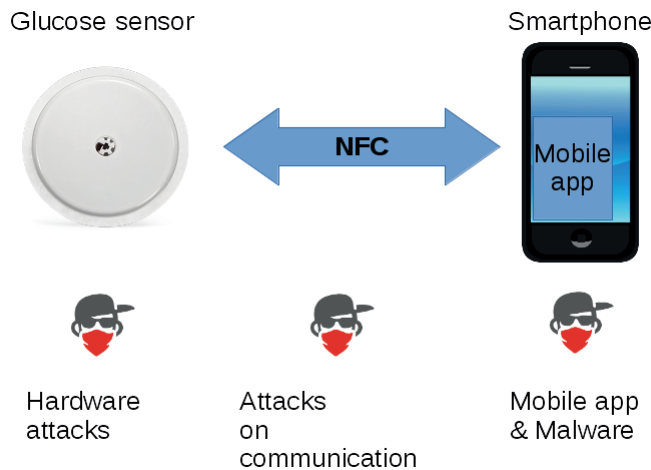


Figure 1: Privacy and security threats on the FGM.

The sensor casing is difficult to open without damage [10]. We developed our own method by unclipping the enzyme sensor and then introducing a thin blade between the upper and lower parts of the casing. Once opened, we identified an RF430 chip, an NFC antenna, a temperature sensor and JTAG test points.

The temperature sensor ensures the sensor is being operated in correct temperature conditions. It probably also serves as a basic hack detection system, detecting whether or not the sensor is being worn on a human body.

NFC communication

There are several NFC flavours. The best known is probably ISO 14443, for instance Mifare cards. Our glucose sensor uses a lesser known model, known as vicinity cards or ISO 15693. A vicinity card can be read at a longer distance than a proximity card, and still does not need to be powered. The reading distance remains short, and while the standard claims up to one metre, diabetic users report that the glucose sensor won't transmit over more than 4cm. Note that this may be due to the NFC reader (or the smartphone acting as an NFC reader). A reader with a larger antenna loop should theoretically be able to read at a longer distance.

This leads us to the first vulnerability: the sensor can be read by anyone. There is no authentication mechanism, no pairing, no device lock. In some cases, this is a useful 'feature': a doctor can scan the glucose sensor of his patient. The downside is that the sensor is open to anyone. In a packed bus, an attacker could scan and read the glucose level of a diabetic person without being noticed. A prying employer or an insurance company would instantly know their employee or client had diabetes, etc. We confirmed the issue by reading the glucose sensor without any problem with HydraBus NFC [11] and Proxmark 3 [12] using, respectively, the commands `scan continuous` and `hf search`.

Reading a glucose measure is more complicated because you need to understand where the measures are located and their format. We reversed the memory map of the sensor with help from the code of open-source mobile applications [13, 14].

A glucose measure consists of a six-byte record. There is one new glucose measure per minute. The 16 most recent records are known as *trend records*. Then, a *historic record* is created. The sensor memorizes 32 historic records. At all times, the sensor keeps track of the index of the current trend record, the index of the current historic record, and the number of minutes since the sensor was activated. The sensor’s memory map is detailed below:

Byte number	Description
26	Current trend index
27	Current history index
28 - 123	Trend records. Six-byte records. 16 records.
124 - 315	History records. Six-byte records. 32 records.
316	Low nibble for sensor time
317	High nibble for sensor time

The format of a glucose record is heavily obfuscated in the official application, so open-source applications interpret them with slight differences, using only the first two bytes. Knowing this, we can read a glucose measure by sending an NFC Read Single Block (0x20) or Read Multiple Blocks (0x23) command.

For example, the command below issues a Read Single Block command to read block number 4. The content of this block is 00 31 82 0F 00 C0 C6 31 82 (the last two bytes, 05 BE, are a CRC). In this case, the glucose record consists of the six bytes, 0F 00 C0 C6 31 82. They indicate a glucose measure of 1.5 mg/dL. This is extremely low because in this case the sensor had been removed from the human body and was no longer measuring glucose.

```
proxmark3> hf 15 cmd raw -c 02 20 04
received 11 octets
00 31 82 0F 00 C0 C6 31 82 05 BE
```

The next attack scenario to test is the modification of glucose measures. Fortunately, this does not work on the sensor because NFC blocks are locked. In the command below, we tried to overwrite block 27. The answer is explicit: this is impossible, the block is locked.

```
proxmark3> hf 15 cmd write u 27 00 00 00 00 78 00 00 00
Tag returned Error 18: The specified block is locked and its content cannot be changed.
```

Note that we tried several methods to overwrite blocks: unaddressed/addressed mode, different block numbers (all were locked) and raw write command. We got the ‘block locked’ error in all cases, so the sensor is *not vulnerable to glucose records modification attacks*.

Finally, as mentioned before, there is no authentication or pairing method between the sensor and the mobile application, and consequently the sensor can be read by anybody. The *opposite* is true

as well: the mobile application cannot check that it is talking to the appropriate sensor. This means that an attacker can create a fake sensor using a Magic ISO 15693 card [15] and write fake glucose entries to the card. The mobile application does not have any way to detect that the sensor is fake. In fact, the official application won't talk to a sensor it hasn't activated, so the attack will only work on a fake sensor with the same UID as the current real sensor. Non-official applications don't have this restriction though. However, if the attack scenario is to have the victim read incorrect glucose values, it is far simpler to implement a fake mobile application, as we will discuss in the malware section.

Mobile application

For mobile applications that communicate with the glucose sensor, we are particularly concerned by privacy leaks. The threats are easy to assess on open-source apps, but more difficult on closed-source ones. We investigated the official app, which fortunately has been implemented with care. We have not witnessed any medical data leak.

The application accesses the Internet in the following circumstances:

- **Licence verification** at startup, using *Google's StrictPolicy*. This ensures the application is the one that comes from the *Play Store*.
- **User login** and access to user profile (via HTTPS). The user profile contains limited information: name, date of birth, email, country, password.
- **Intentional medical data sharing services** (e.g. Diasend).
- **Analytics**. The application uses Firebase to log any event (*Android* activity classes, user interactions with the app, etc.). This is the most disturbing privacy issue we have noted. The information is not sensitive, but the level of detail is worrying. It tracks every single move of the end-user with the application.

```
05-24 08:20:11.384 V/FA      (13108): Event recorded: Event{appId='com.
XXXXXXXXXXXX', name='screen_view(_vs)', params=Bundle[{firebase_event_
origin(_o)=auto, firebase_previous_class(_pc)=SplashActivity, firebase_previous_
id(_pi)=-3985357911052850480, firebase_screen_class(_sc)=HomeActivity, firebase_
screen_id(_si)=-3985357911052850479}]}
```

```
05-24 08:20:11.436 D/FA      (17498): Logging event (FE): SYS_UNEXPECTED,
Bundle[{firebase_event_origin(_o)=app, firebase_screen_class(_sc)=HomeActivity,
firebase_screen_id(_si)=-3985357911052850479}]}
```

```
05-24 08:20:11.526 D/FA      (17498): Logging event (FE): user_engagement(_e),
Bundle[{firebase_event_origin(_o)=auto, engagement_time_msec(_et)=290, firebase_
screen_class(_sc)=HomeActivity, firebase_screen_id(_si)=-3985357911052850479}]}
```

Wear time

The sensor expires after exactly 14 days (before 2018, the limit was 10 days in the US [9]).

We have been able to show that the sensor continues to work after that period. For example, in the following, a sensor aged 14 days 5 hours and 16 minutes is still measuring glucose every minute (the sensor has been removed from the human body, which explains the low glucose values).

```
Trend index: 11
Historic index: 20
Sensor bytes: high=0x4f low=0xfc
Sensor running since 20476 minutes (14 days, 5:16:00)
----
...
Trend record no.11: 2700c0e6ef80 = 3.9 mg/dL
Trend record no.12: 2800c0f2af80 = 4.0 mg/dL
...
```

As an experiment, we dissolved sugar in hot water, poured the solution over a sponge, and pinned the sensor to the sponge. The sensor (now aged 14 days, 5 hours and 23 minutes) clearly works. Initially, the record index was 8: the glucose level goes up to 35.5 mg/dL (record index 10 to 15) and then back down to the original 3.9 mg/dL (after index 15, read index 0 - 2).

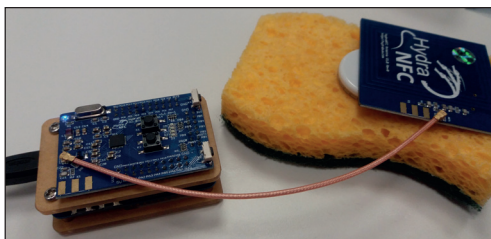


Figure 2: Wet sponge with sugar.

```
Trend index: 2
Historic index: 21
Sensor bytes: high=0x50 low=0x3
Sensor running since 20483 minutes (14 days, 5:23:00)
----
Trend record no. 0: 6f00c0f2d900 = 11.1 mg/dL
Trend record no. 1: 4880c0369c00 = 7.2 mg/dL
Trend record no. 2: 2700c0deaf80 = 3.9 mg/dL
Trend record no. 3: 2700c0eaaaf80 = 3.9 mg/dL
Trend record no. 4: 2600c0f6af80 = 3.8 mg/dL
Trend record no. 5: 2600c0f6af80 = 3.8 mg/dL
Trend record no. 6: 2700c0faaf80 = 3.9 mg/dL
Trend record no. 7: 2600c0eaaaf80 = 3.8 mg/dL
Trend record no. 8: 2600c0f2af80 = 3.8 mg/dL
Trend record no. 9: 2600c002f080 = 3.8 mg/dL
Trend record no.10: 6381802a1580 = 35.5 mg/dL
Trend record no.11: fb8088d29300 = 25.1 mg/dL
Trend record no.12: 9500c07ed400 = 14.9 mg/dL
Trend record no.13: c100c87c1501 = 19.3 mg/dL
Trend record no.14: a000c8b81601 = 16.0 mg/dL
Trend record no.15: 8700c04ed800 = 13.5 mg/dL
```

Sensor expiration is implemented in the mobile application. The code below shows that the application checks that the difference between the current time and the activation date is not greater than the maximum wear time (14 days). This wear time, along with the warm-up time after activation, is not found in the Dalvik executable but read from a native library, `libDataProcessing.so`. This makes it more difficult to hack the values.

```
private SensorEntity addSensorToDatabase(User arg25, String arg26, Object
activationtime, Object arg28, TimeZone arg29, Object current_time_obj,
Object arg31, Object arg32, int arg33, int warmupperiodinminutes, int
weardurationinminutes) throws SensorExpiredException {
    DefaultSensorAbstractionService v1 = this;
    Object activation_time_obj = activationtime;
    int max = weardurationinminutes;
    if (v1.timeOfDayAdapter.toMilliseconds(current_time_obj) -
v1.timeOfDayAdapter.toMilliseconds(activation_time_obj) > TimeUnit.MINUTES.
toMillis(((long)max))) {
        throw new SensorExpiredException();
    }
    ...
}
```

Why 14 days? Why not 15? Why can't patients choose to have some extra time with their sensor if they really need it? The reason is not technical. The sensor could remain operational much longer and the restriction is enforced in software. The real reason is medical... or business-related. From a medical perspective, the device has been certified for 14 days [16], and it is obviously a good idea to remove the sensor after a while and let the skin heal. We leave the business reasons to the reader's imagination.

MALWARE

Another type of attack scenario relies on smartphone compromise.

For example, if an attacker installs malware on a smartphone, s/he will be able to exfiltrate medical data. The malware will need to gain access to the mobile application's private storage (in particular `apollo.db` and `sas.db`), but this is immediate on rooted devices, for instance.

An attacker can also implement a fake mobile glucose application that gives incorrect glucose readings. There is no way to prevent this. The official application does try to ensure that the application is installed from *Google Play* via *Google's* License Verification Library. This will somewhat limit trojanized versions, but not entirely, and malware can be installed from other marketplaces.

An optimist would argue that, besides movie scenarios, there is no motivation for a cybercriminal to create a fake glucose application. We refute this argument, having found *several malicious applications concerning diabetes management*. Their motivation is not really to kill or harm a patient's health, but yet again to make money.

All of the applications shown in Figure 3 are either potentially unwanted apps (e.g. adware), or plain malware:

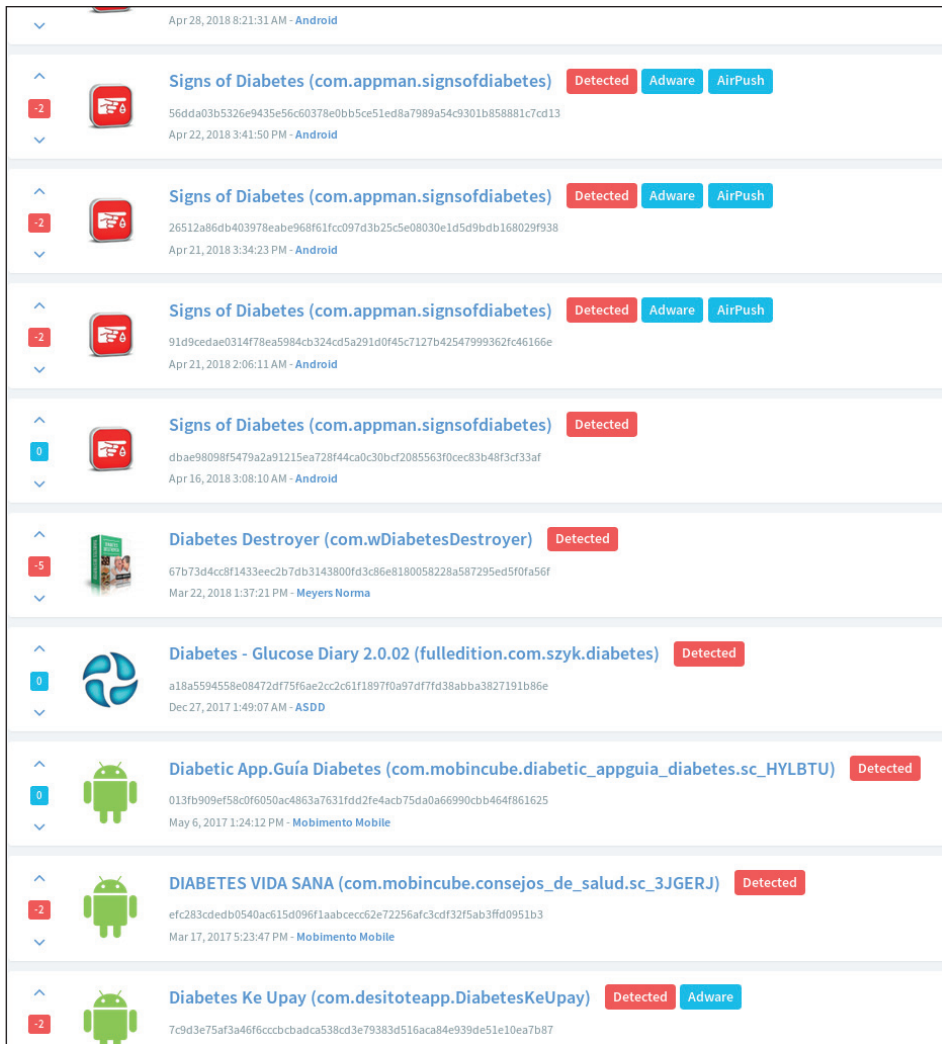


Figure 3: Malicious diabetes management applications for Android smartphones.

- Android/FakeGlucosms.A!tr:** This is an old and typical SMS fraud application. It claims to provide access to a glucose diabetes management application, but only through a paid subscription service, via SMS messages (depositmobi / mobi911). While trying to appear legitimate, nothing is clear in this system: the number of SMSs to be sent depends on the network operator, the user agreement is abusive, SMSs may be resent if the application is ‘activated’ again, etc. Experienced malware analysts will have seen several such cases in the past between 2012 and 2014 [17]. This particular sample also reads a configuration file, from which it will read the URLs of other mobile applications to silently install.


```

public void onCreate(Bundle arg4) {
    super.onCreate(arg4);
    this.actor = new Actor(((Context)this), TextUtils.
getOperatorString(((Context)this)));

    // show link to paid file when more than 2 SMS messages have been sent
    if(this.getSharedPreferences("KEY_PREFS", 0).getInt(TextUtils.
getDailyKey(), 0) > 2) {
        this.showLastScreen(new Intent(((Context)this), ShowURL.class));
    }

    // check airplane mode is not enabled
    if(Settings$System.getInt(this.getContentResolver(), "airplane_mode_on",
0) != 0) {
        this.showAirplaneDialog();
    }
    else {
        // show dialog asking to send SMS messages
        this.setContentView(0x7F030001); // layout:main
        DevReg.sendOpening(((Context)this));
        ...
    }
}

```

- **Riskware/AndroidOSFictus:** This is a legitimate diabetes management application wrapped up in a questionable adware installer. The installer starts by installing the adkit (banners, interstitial ads etc.), then it installs the real diabetes application, and finally hides itself (it does not uninstall itself). The problem is that you won't be able to install the real diabetes application unless you have clicked on the adkit's activity asking you to install 'app of the day' or support their sponsors (see Figure 4).

```

try {
    // get the name of the real diabetes package to install from the assets/
applications directory
    filename = this.getAssets().list("applications")[0];

    // get the filesize of the package
    filesizemb = ((double)(this.getAssets().open("applications/" + filename).
available() / 0x100000));
}
catch(Exception v0) {
    Log.e("tag", v0.getMessage());
}

// InstallTypeActivity is an activity to initiate the adkit
Intent intent = new Intent(((Context)this), InstallTypeActivity.class);

// When the adkit is installed, the riskware will resume and call this class
// which installs the real applications
intent.putExtra("APPWALL_ACTION", "fulledition.com.szyk.diabetes.
KittyFinishActivity");

```

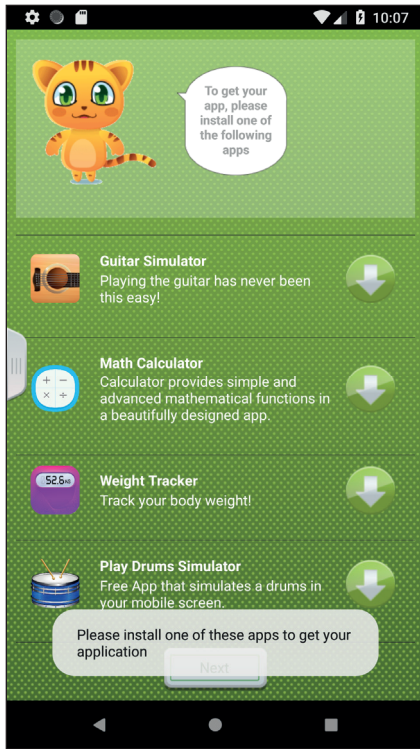


Figure 4: The AndroidOSFictus adware prevents the end-user from installing a diabetes management app until s/he has installed one of the following apps. (Sample SHA256: a18a5594558e08472df75f6ae2cc2c61f1897f0a97df7d38abba3827191b86e.)

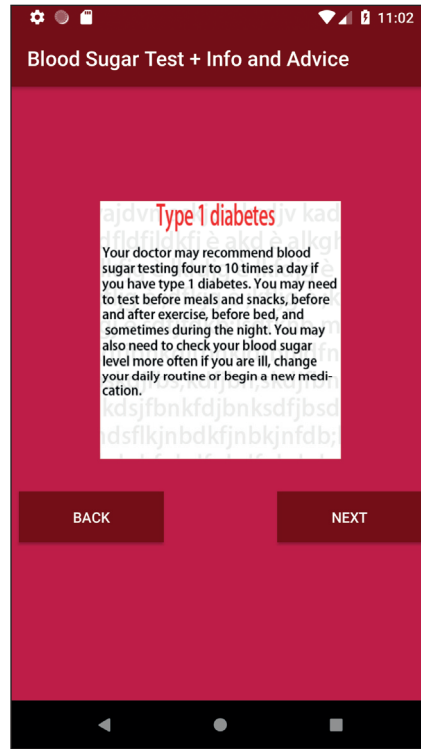


Figure 5: Riskware/Heyzap!Android embeds a diabetes advice application. (Sample SHA256: 34a605a3acbe4e090d2d5c9e308f0834062cce79f348cb7747d28ab49838dea0.)

```
// Messages to display in the adkit
intent.putExtra("PARAMETER_ONE", this.getString(0x7F060027)); //
string:adpath_title_one "You are trying to install"
intent.putExtra("PARAMETER_TWO", filename);
intent.putExtra("PARAMETER_THREE", "(" + String.format("%.2f", Double.
valueOf(filesizemb)) + " MB");
```

- Many other apps are borderline: they do provide information for diabetes but with lots of ads (34a605a3acbe4e090d2d5c9e308f0834062cce79f348cb7747d28ab49838dea0), or they claim to be a diabetes app but do not do anything at all (e.g. 901602a7153b79a0a0e7edf8c8b3b5ca71e95d2a783e2a63e139ce0187aa8312). For example, in between providing a few pieces of diabetes advice, Riskware/Heyzap!Android (Figure 5) regularly shows advertisements and tracks the victim (IP address, GPS location, installed apps) (Figure 6).

3119	53.188071548	172.26.128.23	52.4.5.229	HTTP	651	GET /display?ib=22P02666620277XJ_zMN3vsZeUNUsAzngz6RRvKntDr1Xa312UCUmrnj
3139	53.2140609563	172.26.128.23	23.210.248.44	HTTP	577	GET /t.gif?pid=2087&pidt=0&pidid=83efdb17-1907-4342-ai4a-aba2774ae4e8 HTTP
3165	53.221911093	172.26.128.23	52.4.5.229	HTTP	935	GET /simpleM2M/reportEvent?eventArchetype=impression&requestType=17&event
3215	53.393817016	52.4.5.229	172.26.128.23	HTTP	349	HTTP/1.1 200 OK (GIF89a)
3246	53.452663837	52.1.217.187	172.26.128.23	HTTP	461	HTTP/1.1 302 Found (text/html)
3302	53.550195572	172.26.128.23	143.204.192.127	HTTP	511	GET /analytics/js/ad/2.0/s4m.js HTTP/1.1
3319	53.558709291	172.26.128.23	172.227.178.233	HTTP	570	GET /mnt/InnerActive/Resources/Empty/120x20_empty.gif HTTP/1.1
3351	53.583629105	23.210.248.44	172.26.128.23	HTTP	689	HTTP/1.1 204 No Content
3365	53.614622907	172.227.178.233	172.26.128.23	HTTP	1415	HTTP/1.1 200 OK (GIF89a)
3427	53.652187827	143.204.192.127	172.26.128.23	HTTP	3655	HTTP/1.1 200 OK (application/javascript)
3505	53.770894790	172.26.128.23	93.184.216.34	HTTP	522	GET /favicon.ico HTTP/1.1
3519	53.970173339	93.184.216.34	172.26.128.23	HTTP	1024	HTTP/1.1 404 Not Found (text/html)
12035	87.909191492	172.26.128.23	35.174.74.48	HTTP	227	GET /impression?aid=1170440&t=SCh-RnJKN5DI2Nt_1m100HqemEjBF6GutzxOC1bFH
12047	88.116078892	35.174.74.48	172.26.128.23	HTTP	337	HTTP/1.1 200 OK (GIF89a)
12139	89.609818045	172.26.128.23	23.210.248.44	HTTP	577	GET /t.gif?pid=2037&pidt=0&pidid=1fd57a3e-a899-427a-8a9e-961d7bb3cdc8 HTTP
12156	89.617872487	172.26.128.23	52.1.217.187	HTTP	983	GET /simpleM2M/reportEvent?eventArchetype=impression&requestType=17&event
12157	89.617887041	172.26.128.23	52.4.5.229	HTTP	651	GET /display?ib=22P157846BC38bZ6oqp3KxrlT-e37j0pDl61w5s45dFhc-5BAKzKr
12158	89.617904851	172.26.128.23	35.174.74.48	HTTP	223	GET /impression?aid=1170440&t=jh7mk6Qy2ldkwehve6sva_0Rdg2Z15W5S2TUp]5ft
12203	89.721202737	52.1.217.187	172.26.128.23	HTTP	461	HTTP/1.1 302 Found (text/html)
12207	89.721934623	35.174.74.48	172.26.128.23	HTTP	337	HTTP/1.1 200 OK (GIF89a)
12231	89.808705400	52.4.5.229	172.26.128.23	HTTP	349	HTTP/1.1 200 OK (GIF89a)
12246	90.108708149	172.26.128.23	172.227.178.233	HTTP	671	GET /mnt/InnerActive/Resources/Empty/120x20_empty.gif HTTP/1.1
12257	90.169985143	172.227.178.233	172.26.128.23	HTTP	363	HTTP/1.1 304 Not Modified
12285	90.167248313	23.210.248.44	172.26.128.23	HTTP	689	HTTP/1.1 204 No Content
12421	90.739373272	172.26.128.23	216.58.206.33	HTTP	506	GET /sodar/UFYwWmt.js HTTP/1.1
17443	90.807834767	172.26.128.23	216.58.198.162	HTTP	535	GET /activex/ew/3s/current/11dar_1s2cachez20110914 HTTP/1.1
Request URI Path: /simpleM2M/reportEvent						
Request URI Query [truncated]: eventArchetype=impression&requestType=17&eventType=0&shouldRedirect=false&s=8245406419899997902&aid=600774&cid=						
Request URI Query Parameter: eventArchetype=impression						
Request URI Query Parameter: requestType=17						
Request URI Query Parameter: eventType=0						
Request URI Query Parameter: shouldRedirect=false						
Request URI Query Parameter: s=8245406419899997902						
Request URI Query Parameter: aid=600774						
Request URI Query Parameter: cid=149.5.228.0						
Request URI Query Parameter: ua=Da1v1kK2F2.1.0+%28Linux%3B+Uk%3B+Android+8.1.0%3B+Android+SDK+built+for+x86+Build%2FOSM1.180201.007%29						
Request URI Query Parameter: po=550						
Request URI Query Parameter: cc=FR						
Request URI Query Parameter: adExpTime=1556622429824						
Request URI Query Parameter: adTime=1556618829824						
Request URI Query Parameter: v=2_1_0_heyzap_0_0_0						
Request URI Query Parameter: locationType=3						
Request URI Query Parameter: network=Pubnative_RT_B_Premium						
Request URI Query Parameter: accp=						
00c0	3d 36 30 30 37 37 34 26	63 69 70 3d 31 34 39 2e	=600774&	cid=149.		
00d0	80 2e 32 46 32 2e 30 26	75 61 3d 44 61 6c 76 69	52237&	ua=Da1v1		
00e0	5b 25 32 46 32 2e 31 2e	30 2b 25 32 30 4c 69 6e	kK2F2.1.	0+%28Lin		
00f0	75 78 25 33 42 2b 55 25	33 42 2b 41 6e 64 72 6f	ux%3B+Uk	%3B+Andro		
0100	69 64 2b 38 2e 31 2e 30	25 33 42 2b 41 6e 64 72	id+8.1.0	%3B+Andr		
0110	6f 69 64 2b 53 44 4b 2b	62 75 69 6c 74 2b 66 6f	oid+SDK+	built+fo		
0120	72 2b 78 38 36 2b 42 75	69 6c 64 25 32 4e 4f 53	r+x86+Bu	ild%2FOS		
0130	4d 31 2e 31 38 30 32 30	31 2e 30 30 37 25 32 39	M1.18020	1.007%29		
0140	26 70 6f 3d 35 39 3e 26	63 63 3d 46 52 26 61 64	&po=550&	cc=FR&ad		

Figure 6: Riskware/Heyzap/Android leaking IP address of device (plaintext).

The conclusion is twofold:

1. The good news: so far, we haven't found any malware directly impacting diabetes or a glucose sensor. There is no malware trying to modify glucose readings, no attack/denial of service/exploit of glucose readers or sensors, no malware stealing diabetic measures or related health data to be used or sold by unethical individuals or companies.
2. The bad news: there are many fake applications and adware posing as diabetic management apps, diabetic advice, diabetic stats, nutritional facts, etc. Getting infected by one of these usually does not impact the patient's health, but it uses their medical condition to push unethical advertisement practices. For example, if an end-user finds an application useful to control his/her diabetes, s/he will probably put up with viewing many ads.

Finally, note that mobile ransomware, while not listed in this study, may have a severe impact on a diabetic end-user: the smartphone is locked and cannot be used to monitor blood glucose levels until the phone has been uninfected.

DARKNET

The Darknet is well known for the illegal activities it offers [18, 19]. In fact, not all websites and services on the Darknet are necessarily illegal, but its anonymity, encryption and distributed architecture have attracted many criminals.

For patients with diabetes, this opens up an additional set of threats that we detail below.

Theft of medical records

Medical records are extremely valuable because they contain a person’s name, address, national identity numbers (e.g. Social Security), and medical history. Besides the apparent attack of blackmailing people by threatening to leak their medical records, medical records are valuable to criminals because of other types of attacks they can help facilitate.

One of those types of attacks is identity theft, which is used to open bank accounts and apply for credit cards and loans. Buying stolen credit cards is becoming difficult for attackers because the credit card industry is getting very good at finding and stopping fraudulent transactions. However, stolen medical records have all the information an attacker may need to steal someone’s identity, get credit cards and loans in their name. Stolen medical records typically sell for \$103 per record (compared to \$3 per record for a stolen credit card). Online marketplaces have a significantly higher price, but online sellers typically negotiate a lower price.

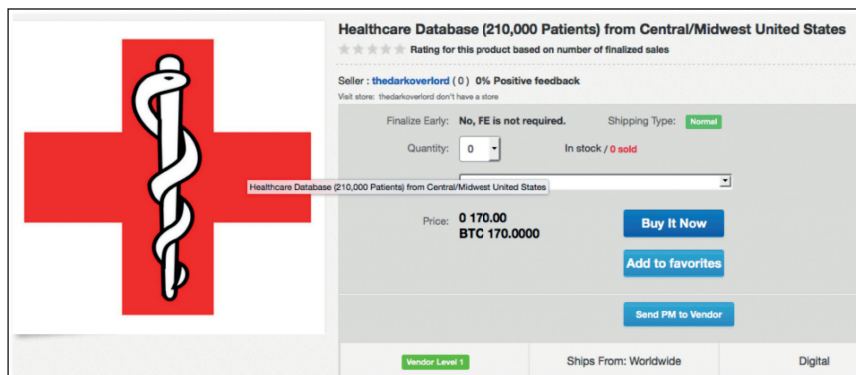


Figure 7: Healthcare database for sale on mirror of Wall Street marketplace, in 2018.

Attackers will typically claim they are breaking into organizations that have access to medical tracking software. In October 2018, a user of thedarkoverlord posted a picture of a stolen medical record database as evidence that he had stolen medical records in his possession.

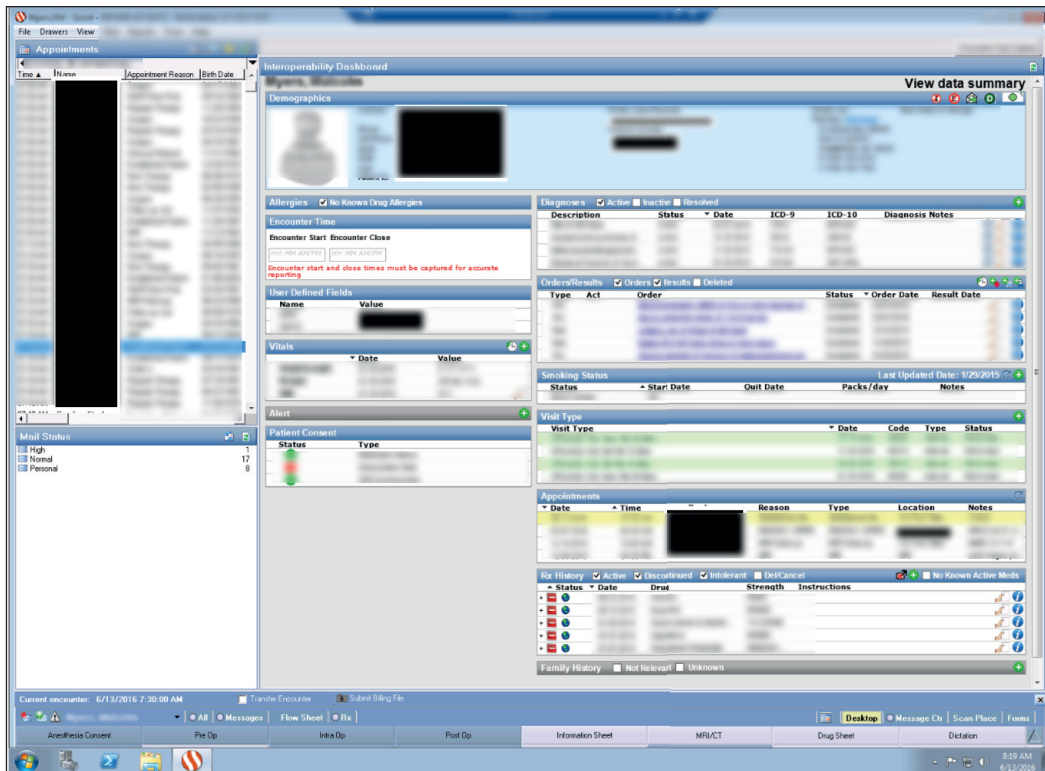


Figure 8: Medical records for sale on thedarkoverlord, October 2018.

The username, the marketplace and item are no longer available for sale so we cannot confirm if this is correct. However, the data does appear to be from a medical office.


Medicine



Users attempting to buy medicine and other drugs from the Darknet is nothing new. Some have estimated that the Darknet is the most prominent commerce location available for drugs in most countries. A casual user will quickly come across multiple sites that are selling drugs. A new wave of prescription drugs is being purchased illegally by users who cannot afford the medicine through traditional and legal pharmacies, but who desperately need it.

In some cases, the medicine is purchased cheaply because of connections or discounts from the sellers. In other cases, the medicine may be cheaper in a different country but has been illegally exported. *Lantus* is a popular insulin supplemental medicine. According to *Pharmacy Times*, the typical cost for this and similar medicine is around \$400. The same medicine, in the same dosage, is being sold on the Darknet for \$149 (Figure 9).

Humalog Kwipen [20] is an insulin device given to people who have type 2 diabetes. At the time writing, the authors of this paper checked with five national insurance companies, including those

Lantus Solostar - Insulin - 100IU 10x3ml - Sanofi + Add Listing



Sold by  pharmamed 2 5.00  Trust Level 1

FEATURES			
Product class	Physical Package	Quantity left	Unlimited
Ships from	European Union		
Ships to	WorldWide		
Views	12	Visibility	Public
Ends In	Never	Payment	Escrow

Unit price: USD 149.60 0.01768093 BTC

Europe 3 Days - USD +11.51 / Item

Buy Now


Maximum Quantity: Unlimited



Feedback	Buyer/Price	Date/Time	Rating
This product doesn't currently have any feedback.			

Figure 9: Insulin supplement sold at low cost on Nightmare, May 2019.

offered under the United States Affordable Healthcare Act (commonly referred to as ObamaCare), and each of them covered the medicine for patients in the United States for approximately \$95. Observe in Figure 10 the medicine is being sold on the Darknet for approximately \$92. One can observe this medicine is primarily being advertised to buyers from the United States. In all likelihood, no one is going to buy this medicine from an illegal source just to save a few dollars. However, for those who have no healthcare insurance, the cost of the medicine is around \$360. According to [21], there are 44 million people in the United States that do not have any form of health insurance. There are also an additional 38 million people who have inadequate insurance. It is crucial to remember that users of this medicine need an ongoing supply for the rest of their lives. If they do not take it, they will be taking a significant and perhaps even fatal risk.

Humalog KwikPen - Insulin - 100IUx5x3ml - Lilly + Add Listing



Sold by  pharmamed 2 5.00  Trust Level 1

FEATURES			
Product class	Physical Package	Quantity left	Unlimited
Ships from	European Union		
Ships to	WorldWide		
Views	13	Visibility	Public
Ends In	Never	Payment	Escrow

Unit price: USD 92.06 0.01088057 BTC

Europe 3 Days - USD +11.51 / Item

Buy Now

Maximum Quantity: Unlimited

Feedback	Buyer/Price	Date/Time	Rating
This product doesn't currently have any feedback.			

Figure 10: Humalog Kwikpen (fast-acting insulin) sold at a bargain price for patients with no healthcare insurance.

We also found several other diabetes-related medicines for sale: Gabapentin pills used for the treatment of neuropathic pain caused by diabetic neuropathy, Metformin to decrease the amount of sugar the intestine absorbs and make the body more sensitive to insulin, etc.



Figure 11: Gabapentin, typically used to treat neuropathic pain.

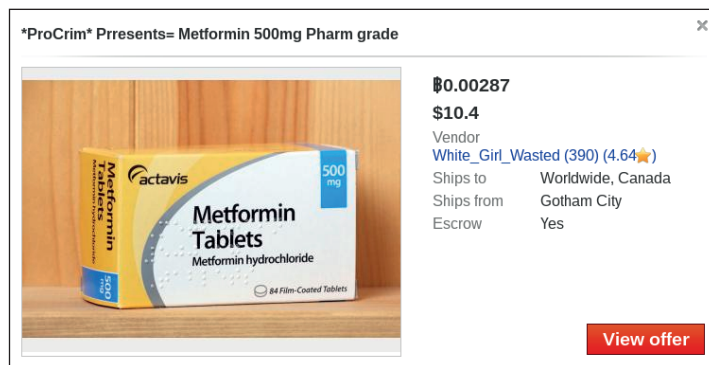


Figure 12: Other diabetic treatments found on the DreamMarket, January 2019.

Medicine sold for more complicated treatment is often more difficult to find. *Invokana* is advertised as the only type 2 diabetes medicine approved by the United States government to help lower A1C and reduce the risk of major cardiovascular events such as strokes and heart attacks. The retail price of this medicine can easily reach over \$600. The screenshot shown in Figure 13, which was taken on 29 May 2019 from the Silk Roads 3.0 marketplace, shows the medicine being sold for \$150 to \$290 (price varies depending on dosage).

Of course, one of the things we do not know is whether this medicine is real. As reported in [22], often the medicine is legitimate and being sold after having been purchased in a country that has lower prices, or it may have been stolen or obtained through other means of fraud. Yet, there have been several cases where the medicine quality was reduced, mixed in with other products, or even fake. In February 2019, [23] reported that school pupils from Sussex in the UK purchased anti-anxiety pills which probably also contained Fentanyl, a powerful but dangerous painkiller.

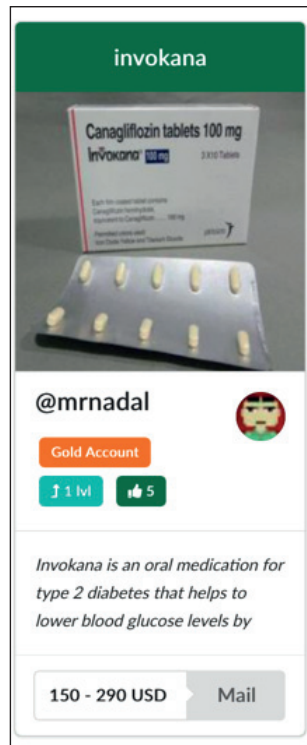


Figure 13: Invokana sold on the Silk Roads 3.0 marketplace.

CONCLUSION

The security and privacy of IoT is often close to non-existent. On investigating this connected glucose sensor it came as a nice surprise to find clear signs that the vendor had paid attention to this aspect. We summarize the major issues we found:

- Anybody can read the records from the connected glucose sensor. In some situations, the mere fact of wearing a glucose sensor for diabetes can be considered as sensitive information. This is a design issue as there is no authentication/pairing mechanism. On the other hand, it is probably important to keep the sensor readable by doctors/nurses treating the patient.
- The official application stops using the sensor after exactly 14 days, whereas the sensor could be used for a longer period. There are medical reasons for limiting the wear time, nonetheless this choice looks close to planned obsolescence when the patient could choose to get extra time, or the sensor could be packaged just to change the soft needle.
- End-users need to be cautious about the source from which they download their diabetes applications because malicious diabetes applications exist (not just one or two, but several). The malicious apps we have found so far do not impact the victim's health, but they use diabetes as a means to get trusted and installed on the smartphone.

- There is evidence of medical records being sold underground, in particular for identity theft. This issue will affect everyone, but a full, detailed, medical record of a diabetic patient is likely to sell at a higher price because of its freshness.
- Lots of diabetic treatments are sold underground, usually at cheaper prices. While it appears the medicine is often real, diabetic patients face the risk of reduced quality, or even poisoning.

ACKNOWLEDGEMENTS

For this research, we received lots of support from other researchers that we wish to thank. First, we thank several diabetic contacts who helped us understand how they cope with their diabetes and how they use FGMs. We withhold their names, but express our deepest gratitude for the time they spent answering our questions, providing data from their sensors, and even supplying a few used sensors for our tests.

We also thank several researchers who helped us with various parts of our research: Ludovic Apvrille (opening the sensor), Philippe Paget (soldering), Philippe Teuwen (NFC questions), pancake (reversing issues), Aurélien Francillon (equipment), Nicolas Oberli (HydraNFC) and Iceman (proxmark3 questions).

REFERENCES

- [1] BBC News. Three US hospitals hit by ransomware. <https://www.bbc.com/news/technology-35880610>.
- [2] Infosec Institute. Ransomware Case Studies: Hollywood Presbyterian & The Ottawa Hospital. <https://resources.infosecinstitute.com/category/healthcare-information-security/healthcare-attack-statistics-and-case-studies/ransomware-case-studies-hollywood-presbyterian-and-the-ottawa-hospital/>.
- [3] Siwicki, B. 71% of IoT medical device ransomware infections caused by user practice issues. <https://www.healthcareitnews.com/news/71-iot-medical-device-ransomware-infections-caused-user-practice-issues>.
- [4] Davis, J. Ransomware Attack on May Eye Care Breaches 30K Patient Records. <https://healthitsecurity.com/news/ransomware-attack-on-may-eye-care-breaches-30k-patient-records>.
- [5] Regalado, D. Discovery of Cyberattack Trends Targeting Connected Medical Device. http://go.zingbox.com/rs/562-ZPO-907/images/Zingbox_Medical_Device_Cyberattack_Trend_Report.pdf.
- [6] Ping Lew, K. Cryptomining and IoT attacks gunning for healthcare industry. <https://securitybrief.co.nz/story/cryptomining-and-iot-attacks-gunning-healthcare-industry>.
- [7] World Health Organization. Diabetes. https://www.who.int/dietphysicalactivity/media/en/gsf_s_diabetes.pdf.
- [8] Cengiz, E.; Tamborlane, W.V. A Tale of Two Compartments: Interstitial Versus Blood Glucose Monitoring. *Diabetes Technology & Therapeutics Journal*, June 2009. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2903977/>.

- [9] Kwon, J.; Levine, B.; Brow, A. FreeStyle Libre in US Now Approved for 14-Day War and 1 Hour Warmup. <https://diatribe.org/freestyle-libre-us-now-approved-14-day-wear-and-1-hour-warmup>.
- [10] Langley, S. FreeStyle Libre Glucose Sensor Tear Apart? <https://www.youtube.com/watch?v=40RXFhZp8hg>.
- [11] HydraNFC 1.0 Specifications. <https://hydrabus.com/hydranfc-1-0-specifications/>.
- [12] Proxmark 3. <https://github.com/Proxmark/proxmark3>.
- [13] Vicktor. <https://github.com/vicktor/FreeStyleLibre-NFC-Reader>.
- [14] Liapp. The unofficial Android app for the Abbott Freestyle Libre. <https://github.com/CMKlug/Liapp>.
- [15] 15693 UID Changeable + Lua Script by Iceman. <https://www.rfxsecure.com/product/icode-sli-icode-slix-uid-custom-inlay/>.
- [16] Bailey, T.; Bode, B.; Christiansen, M.; Klaff, L.; Alva, S. The Performance and Usability of a Factory-Calibrated Flash Glucose Monitoring System. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4649725/>, November 2015, Journal of Diabetes Technology & Therapeutics.
- [17] Goujon, A.; Ramos, P. Boxer SMS Trojan. https://www.welivesecurity.com/wp-content/media_files/SMS_Trojan_Whitepaper.pdf.
- [18] Lovet, G. Dirty Money on the Wires: The Business Models of Cyber Criminals, Virus Bulletin 2006. <https://fortiguard.com/events/718/2006-09-01-dirty-money-on-the-wires-the-business-models-of-cyber-criminals>.
- [19] Balduzzi, M.; Ciancaglini, V. Cybercrime in the Deep Web, Black Hat EU. <https://www.blackhat.com/docs/eu-15/materials/eu-15-Balduzzi-Cybercrime-In-The-Deep-Web-wp.pdf>.
- [20] Humalog Kwikpen. <https://www.humalog.com/u-200-kwikpen/>.
- [21] Healthcare crisis. The uninsured. <https://www.pbs.org/healthcarecrisis/uninsured.html>.
- [22] Krebs, B. Spam Nation: The Inside Story of Organized Cybercrime – from Global Epidemic to Your Front Door, 2014.
- [23] BBC News. Fake Xanax pills sold to pupils ‘may contain Fentanyl’. February 2019. <https://www.bbc.co.uk/news/uk-england-sussex-47250849>.