ROCK APPROUND THE CLOCK!

TRACKING MALWARE DEVELOPERS BY ANDROID "AAPT" TIMEZONE DISCLOSURE BUG

@unapibageek - @ssantosv

Sheila Ayelen Berta

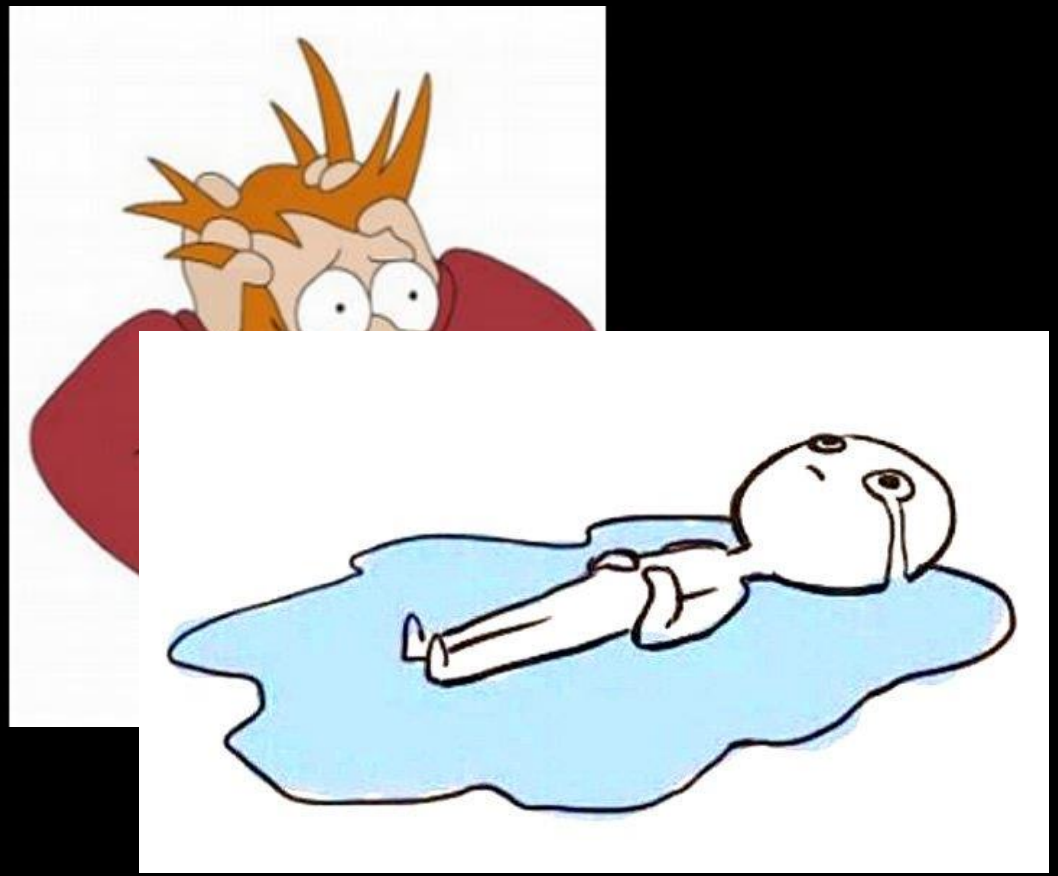
Security Researcher
ElevenPaths
(Telefonica Digital cyber security unit)

Sergio De Los Santos

Head of Innovation and Lab
ElevenPaths
(Telefonica Digital cyber security unit)

@unapibageek - @ssantosv

@unapibageek - @ssantosv



WHAT WE DID?

WHAT WE DID?

ANDROID

@unapibageek - @ssantosv

WHAT WE DID?

@unapibageek - @ssantosv

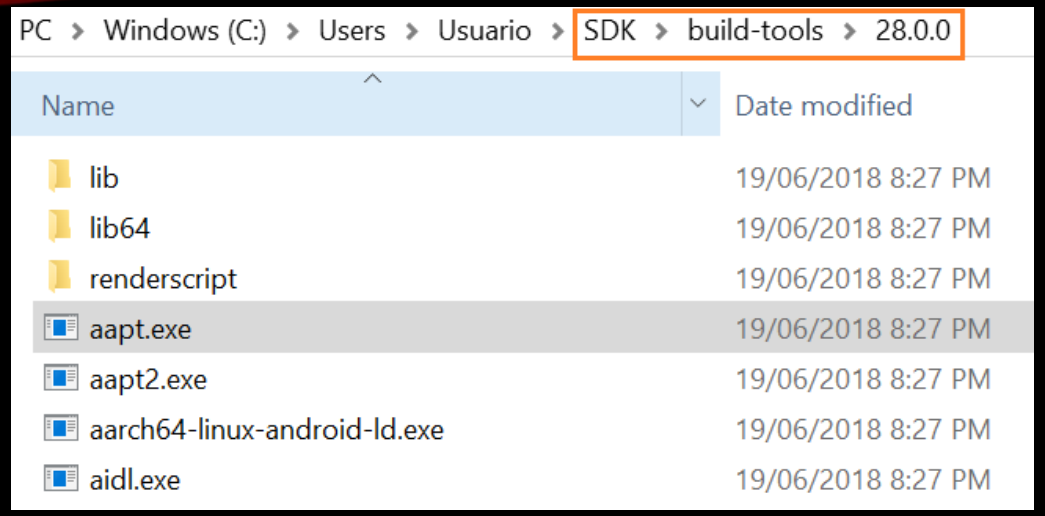
WHAT WE DID?

@unapibageek - @ssantosv

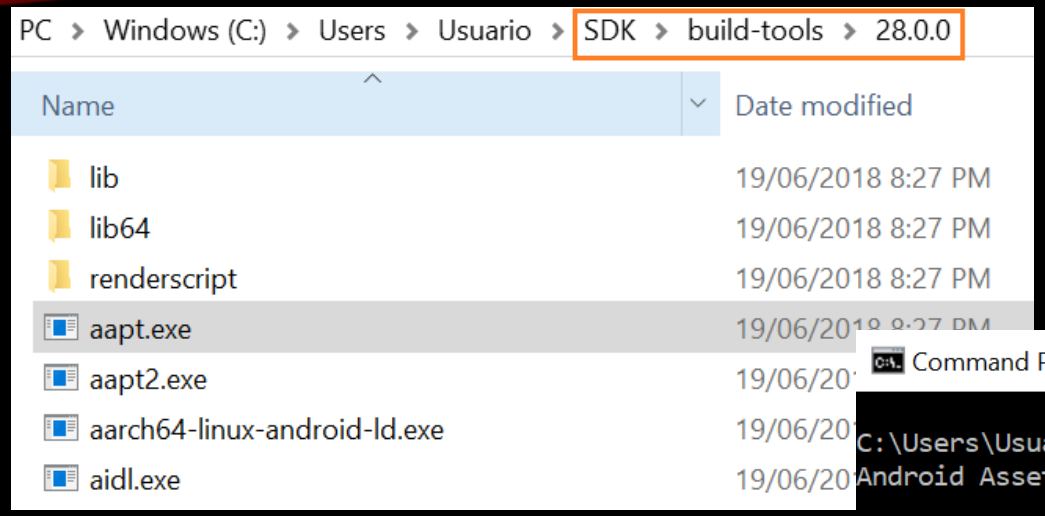
ANDROID

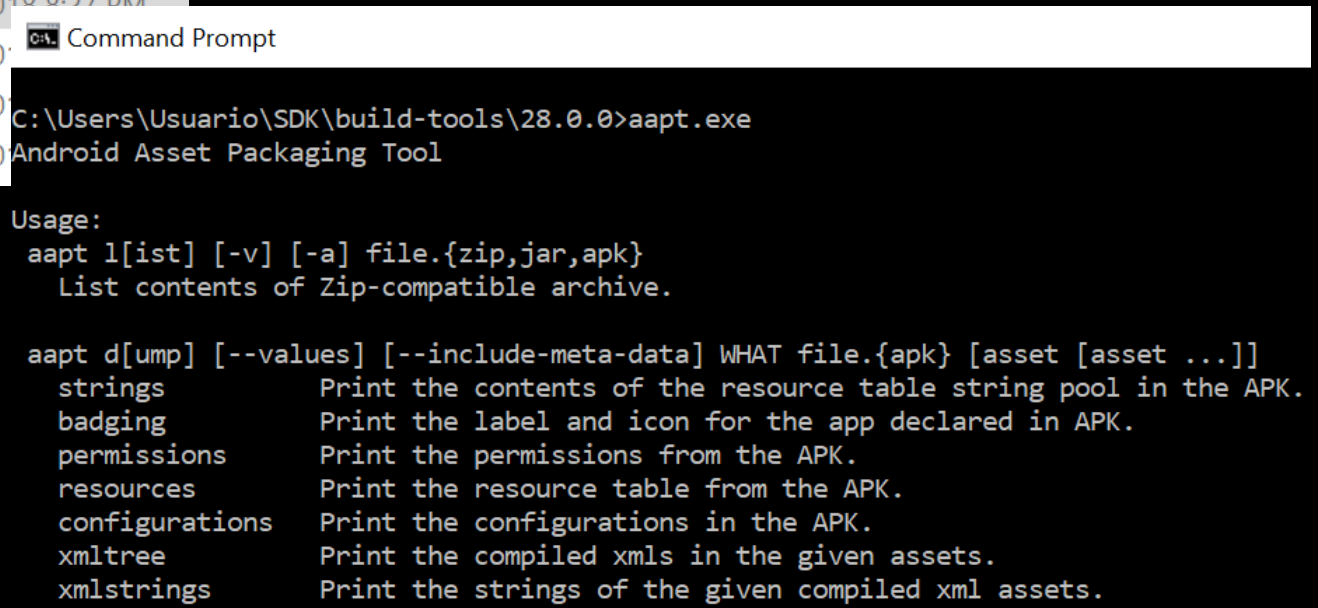
@unapibageek - @ssantosv

WHAT IS AAPT?


PC › Windows (C:) › Users › Usuario › SDK › build-tools › 28.0.0
Name
Date modified
lib 19/06/2018 8:27 PM
lib64 19/06/2018 8:27 PM
renderscript 19/06/2018 8:27 PM
aapt.exe 19/06/2018 8:27 PM
aapt2.exe 19/06/2018 8:27 PM
aarch64-linux-android-ld.exe 19/06/2018 8:27 PM
aidl.exe 19/06/2018 8:27 PM

WHAT IS AAPT?





@unapibageek - @ssantosv

AAPT TIMEZONE DISCLOSURE

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe add TEST01.apk AndroidManifest.xml test.txt
 'AndroidManifest.xml'...
 'test.txt'...

C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TEST01.apk
Archive:  TEST01.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  01-01-80 03:00  ef681d41  AndroidManifest.xml
       4  Stored        4   0%        61  01-01-80 03:00  6fa0f988  test.txt
--------          -------  ---                            -------
      17               16   6%                         2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

@unapibageek - @ssantosv

AAPT TIMEZONE DISCLOSURE

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe add TEST01
 'AndroidManifest.xml'...
 'test.txt'...

C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TEST0
Archive:  TEST01.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CR      ----
--------  ------  -------  -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  01-01-80 03:00  ef681d41  AndroidManifest.xml
       4  Stored        4   0%        61  01-01-80 03:00  6fa0f988  test.txt
--------          -------  ---                            -------
      17               16   6%                         2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

```
     Date   Time
     ----   ----
01-01-80 03:00
01-01-80 03:00
```

@unapibageek - @ssantosv

AAPT TIMEZONE DISCLOSURE

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe add TEST01
 'AndroidManifest.xml'...
 'test.txt'...

C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TEST0
Archive:  TEST01.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CR      ----
--------  ------  -------  -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  01-01-80 03:00  ef681d41  AndroidManifest.xml
       4  Stored        4   0%        61  01-01-80 03:00  6fa0f988  test.txt
--------          -------  ---                            -------
      17               16   6%                         2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

```
     Date   Time
     ----   ----
01-01-80 03:00
01-01-80 03:00
```

```
     Date   Time
     ----   ----
01-01-80 08:00
01-01-80 08:00
```

@unapibageek - @ssantosv

AAPT TIMEZONE DISCLOSURE

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe add TEST01
 'AndroidManifest.xml'...
 'test.txt'...

C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TEST0
Archive:  TEST01.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CR      ----
--------  ------  -------  -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%          0  01-01-80 03:00  ef681d41  AndroidManifest.xml
       4  Stored        4   0%         61  01-01-80 03:00  6fa0f988  test.txt
--------          -------  ---                            -------
      17               16   6%                         2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

```
     Date   Time
     ----   ----
01-01-80 03:00
01-01-80 03:00
```

```
     Date   Time
     ----   ----
01-01-80 04:00
01-01-80 04:00
```

```
     Date   Time
     ----   ----
01-01-80 08:00
01-01-80 08:00
```

@unapibageek - @ssantosv

ANALYZING SOURCE CODE

Google Git

android / platform / frameworks / base.git / master / . / tools / **aapt**

ANALYZING SOURCE CODE

Google Git

android / platform / frameworks / base.git / master / . / tools / **aapt**

- ZipEntry.cpp
- ZipEntry.h
- ZipFile.cpp
- ZipFile.h

ANALYZING SOURCE CODE

Google Git

android / platform / frameworks / base.git / master / . / tools / **aapt**

- ZipEntry.cpp
- ZipEntry.h
- ZipFile.cpp
- ZipFile.h

```
358  status_t ZipFile::addCommon(const char* fileName, const void* data, size_t size,
359      const char* storageName, int sourceType, int compressionMethod,
360      ZipEntry** ppEntry)
361  {
```

@unapibageek - @ssantosv

ANALYZING SOURCE CODE

```
498         pEntry->setDataInfo(uncompressedLen, endPosn - startPosn, crc,
499             compressionMethod);
500         pEntry->setModWhen(modWhen);
501         pEntry->setLFHOffset(lfhPosn);
```

ANALYZING SOURCE CODE

```
498         pEntry->setDataInfo(uncompressedLen, endPosn - startPosn, crc,
499             compressionMethod);
500         pEntry->setModWhen(modWhen);
501         pEntry->setLFHOffset(lfhPosn);
```

```
366         unsigned long crc;
367         time_t modWhen = 0;
368
```

ANALYZING SOURCE CODE

ANALYZING SOURCE CODE

ZipEntry.cpp
ZipEntry.h

```
340  void ZipEntry::setModWhen(time_t when)
341  {
```

ANALYZING SOURCE CODE

ZipEntry.cpp
ZipEntry.h

```
340  void ZipEntry::setModWhen(time_t when)
341  {
```

```
345      time_t even;
346      unsigned short zdate, ztime;
347
348      struct tm* ptm;
349
350      /* round up to an even number of seconds */
351      even = (time_t)(((unsigned long)(when) + 1) & (~1));
352
```

@unapibageek - @ssantosv

ANALYZING SOURCE CODE

ZipEntry.cpp
ZipEntry.h

```
340  void ZipEntry::setModWhen(time_t when)
341  {
```

```
345      time_t even;
346      unsigned short zdate, ztime;
347
348      struct tm* ptm;
349
350      /* round up to an even number of seconds */
351      even = (time_t)(((unsigned long)(when) + 1) & (~1));
352
```

EVEN = 0

ANALYZING SOURCE CODE

```
354   #if !defined(_WIN32)
355       ptm = localtime_r(&even, &tmResult);
356   #else
357       ptm = localtime(&even);
358   #endif
```

ANALYZING SOURCE CODE

```
354   #if !defined(_WIN32)
355       ptm = localtime_r(&even, &tmResult);
356   #else
357       ptm = localtime(&even);
358   #endif
```

```
360       int year;
361       year = ptm->tm_year;
362       if (year < 80)
363           year = 80;
364
365       zdate = (year - 80) << 9 | (ptm->tm_mon+1) << 5 | ptm->tm_mday;
366       ztime = ptm->tm_hour << 11 | ptm->tm_min << 5 | ptm->tm_sec >> 1;
367
368       mCDE.mLastModFileTime = mLFH.mLastModFileTime = ztime;
369       mCDE.mLastModFileDate = mLFH.mLastModFileDate = zdate;
370   }
```

THE PROBLEM

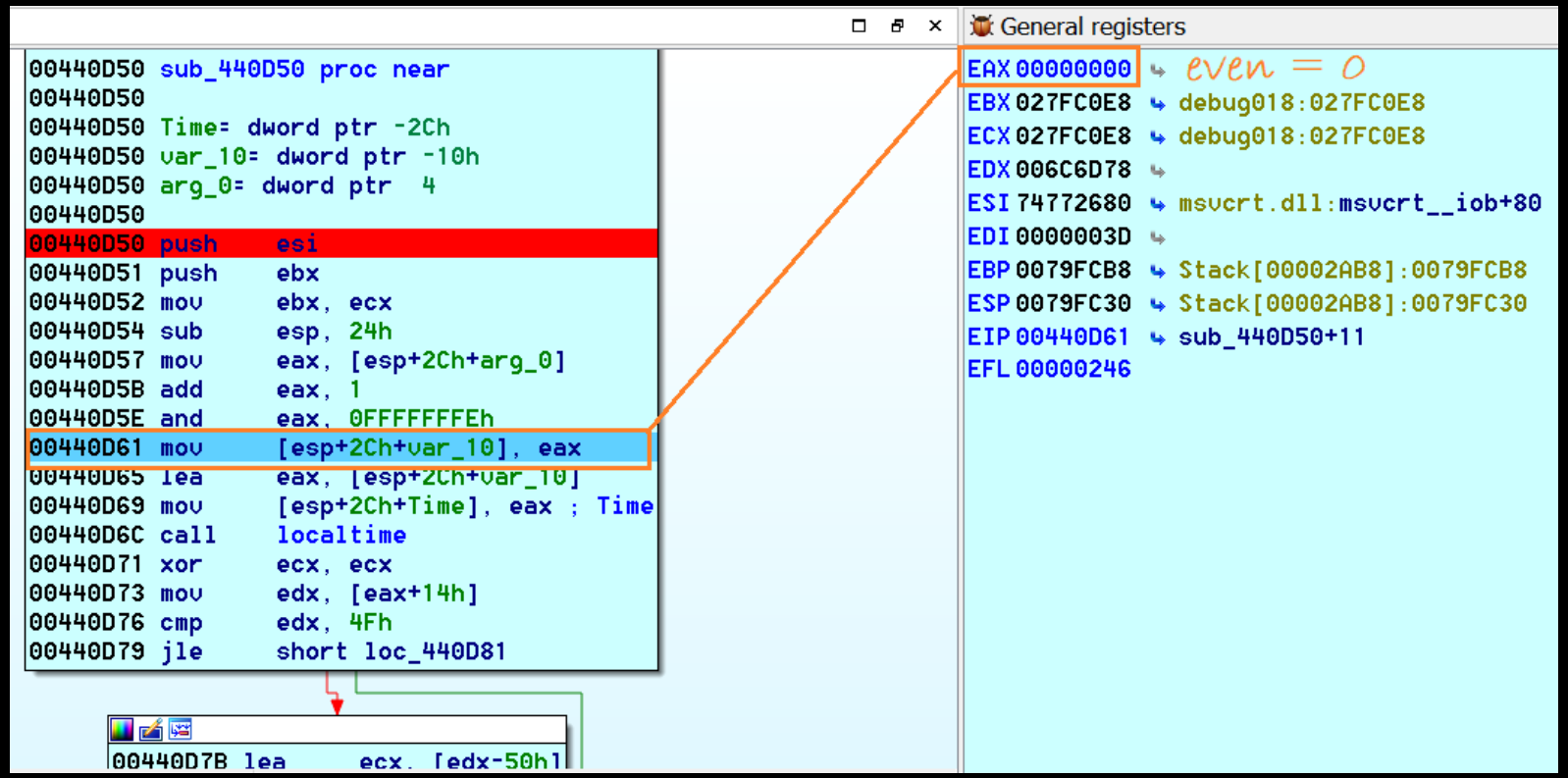
BUG = LOCALTIME(0)

THE PROBLEM

BUG = LOCALTIME(0)

EXPECTED = LOCALTIME(TIMESTAMP)

RUNTIME ANALYSIS



@unapibageek - @ssantosv

RUNTIME ANALYSIS

```
00440D50 sub_440D50 proc near
00440D50
00440D50 Time= dword ptr -2Ch
00440D50 var_10= dword ptr -10h
00440D50 arg_0= dword ptr  4
00440D50
00440D50 push    esi
00440D51 push    ebx
00440D52 mov     ebx, ecx
00440D54 sub     esp, 24h
00440D57 mov     eax, [esp+2Ch+arg_0]
00440D5B add     eax, 1
00440D5E and     eax, 0FFFFFFFEh
00440D61 mov     [esp+2Ch+var_10], eax
00440D65 lea     eax, [esp+2Ch+var_10]
00440D69 mov     [esp+2Ch+Time], eax ; Time
00440D6C call    localtime
00440D71 xor     ecx, ecx
00440D73 mov     edx, [eax+14h]
00440D76 cmp     edx, 4Fh
00440D79 jle     short loc_440D81
```

```
00440D7B lea     ecx, [edx-50h]
```

General registers

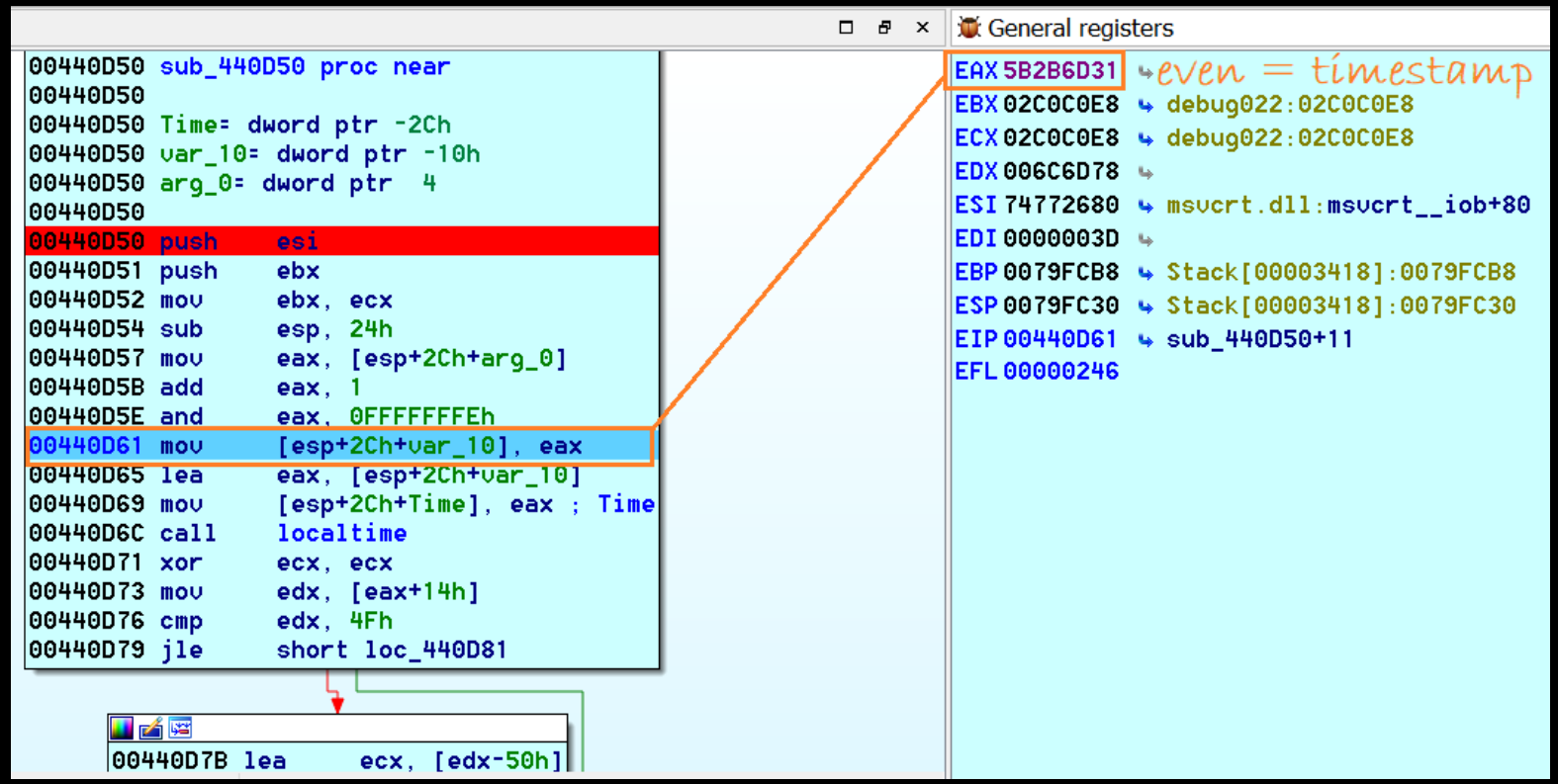
```
EAX 00000000  even = 0
EBX 027FC0E8  debug018:027FC0E8
ECX 027FC0E8  debug018:027FC0E8
EDX 006C6D78
ESI 74772680  msvcrt.dll:msvcrt__iob+80
EDI 0000003D
EBP 0079FCB8  Stack[00002AB8]:0079FCB8
ESP 0079FC30  Stack[00002AB8]:0079FC30
EIP 00440D61  sub_440D50+11
EFL 00000246
```

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TESTGMT02.apk
Archive:  TESTGMT02.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  01-01-80 03:00  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  01-01-80 03:00  00000000  test.txt
--------          -------  ---                                     -------
      13               12   8%                                     2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

RUNTIME ANALYSIS



@unapibageek - @ssantosv

RUNTIME ANALYSIS

```
00440D50 sub_440D50 proc near
00440D50
00440D50 Time= dword ptr -2Ch
00440D50 var_10= dword ptr -10h
00440D50 arg_0= dword ptr  4
00440D50
00440D50 push    esi
00440D51 push    ebx
00440D52 mov     ebx, ecx
00440D54 sub     esp, 24h
00440D57 mov     eax, [esp+2Ch+arg_0]
00440D5B add     eax, 1
00440D5E and     eax, 0FFFFFFFEh
00440D61 mov     [esp+2Ch+var_10], eax
00440D65 lea     eax, [esp+2Ch+var_10]
00440D69 mov     [esp+2Ch+Time], eax ; Time
00440D6C call    localtime
00440D71 xor     ecx, ecx
00440D73 mov     edx, [eax+14h]
00440D76 cmp     edx, 4Fh
00440D79 jle     short loc_440D81
```

```
00440D7B lea     ecx, [edx-50h]
```

General registers

```
EAX 5B2B6D31  even = timestamp
EBX 02C0C0E8  debug022:02C0C0E8
ECX 02C0C0E8  debug022:02C0C0E8
EDX 006C6D78
ESI 74772680  msvcrt.dll:msvcrt__iob+80
EDI 0000003D
EBP 0079FCB8  Stack[00003418]:0079FCB8
ESP 0079FC30  Stack[00003418]:0079FC30
EIP 00440D61  sub_440D50+11
EFL 00000246
```

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TESTGMT03.apk
Archive:  TESTGMT03.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  06-21-18 12:17  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  06-21-18 12:17  00000000  test.txt
--------          -------  ---                                     -------
      13               12   8%                                     2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

RUNTIME ANALYSIS

```
00440D50 sub_440D50 proc near
00440D50
00440D50 Time= dword ptr -2Ch
00440D50 var_10= dword ptr -10h
00440D50 arg_0= dword ptr  4
00440D50
00440D50 push    esi
00440D51 push    ebx
00440D52 mov     ebx, ecx
00440D54 sub     esp, 24h
00440D57 mov     eax, [esp+2Ch+arg_0]
00440D5B add     eax, 1
00440D5E and     eax, 0FFFFFFFEh
00440D61 mov     [esp+2Ch+var_10], eax
00440D65 lea     eax, [esp+2Ch+var_10]
00440D69 mov     [esp+2Ch+Time], eax ; Time
00440D6C call    localtime
00440D71 xor     ecx, ecx
00440D73 mov     edx, [eax+14h]
00440D76 cmp     edx, 4Fh
00440D79 jle     short loc_440D81
```

```
00440D7B lea     ecx, [edx-50h]
```

General registers

```
EAX 5B2B6D31  even = timestamp
EBX 02C0C0E8  debug022:02C0C0E8
ECX 02C0C0E8  debug022:02C0C0E8
EDX 006C6D78
ESI 74772680  msvcrt.dll:msvcrt__iob+80
EDI 0000003D
EBP 0079FCB8  Stack[00003418]:0079FCB8
ESP 0079FC30  Stack[00003418]:0079FC30
EIP 00440D61  sub_440D50+11
EFL 00000246
```

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v TESTGMT03.apk
Archive:  TESTGMT03.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  06-21-18 12:17  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  06-21-18 12:17  00000000  test.txt
--------          -------  ---                                     -------
      13               12   8%                                     2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

@unapibageek - @ssantosv

WHY A TIMEZONE?

Localtime():

HOUR = UNIX EPOCH +/- TIMEZONE

WHY A TIMEZONE?

Localtime():

HOUR = UNIX EPOCH +/- TIMEZONE

E.g:

GMT +3 = UNIX EPOCH + 3Hs

GMT -3 = UNIX EPOCH - 3Hs

WHY A TIMEZONE?

AAPT BUG = 0 +/- TIMEZONE

WHY A TIMEZONE?

AAPT BUG = 0 +/- TIMEZONE

E.g:

GMT +3 = 0+3 = 01-01-80 03:00

GMT -3 = 0-3 = 12-31-79 21:00

A LITTLE DETAIL

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>tzutil /g
Argentina Standard Time
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v GMT-3.apk
Archive:  GMT-3.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  12-31-80 21:00  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  12-31-80 21:00  00000000  test.txt
--------          -------  ---                            -------
      13               12   8%                            2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

@unapibageek - @ssantosv

A LITTLE DETAIL

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>tzutil /g
Argentina Standard Time
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v GMT-
Archive:  GMT-3.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  12-31-80 21:00  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  12-31-80 21:00  00000000  test.txt
--------          -------  ---                            -------
      13               12   8%                            2 files

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

```
      Date   Time
      ----   ----
12-31-80 21:00
12-31-80 21:00
```

@unapibageek - @ssantosv

A LITTLE DETAIL

Command Prompt

```
C:\Users\Usuario\SDK\build-tools\28.0.0>tzutil /g
Argentina Standard Time
C:\Users\Usuario\SDK\build-tools\28.0.0>aapt.exe l -v GMT-
Archive:  GMT-3.apk
 Length   Method    Size  Ratio   Offset      Date  Time  CRC-32    Name
--------  ------  ------- -----  -------      ----  ----  ------    ----
      13  Deflate      12   8%         0  12-31-80 21:00  ef681d41  AndroidManifest.xml
       0  Stored        0   0%        61  12-31-80 21:00  00000000  test.txt
--------          -------  ---
      13               12   8%

C:\Users\Usuario\SDK\build-tools\28.0.0>
```

```
      Date   Time
      ----   ----
12-31-80 21:00
12-31-80 21:00
```

```
360         int year;
361         year = ptm->tm_year;
362         if (year < 80)
363             year = 80;
364
365         zdate = (year - 80) << 9 | (ptm->tm_mon+1) << 5 | ptm->tm_mday;
366         ztime = ptm->tm_hour << 11 | ptm->tm_min << 5 | ptm->tm_sec >> 1;
367
368         mCDE.mLastModFileTime = mLFH.mLastModFileTime = ztime;
369         mCDE.mLastModFileDate = mLFH.mLastModFileDate = zdate;
```

CORRECTION FACTOR!!

OFFSET TABLE

GMT +0 = 01-01-80 00:00
GMT +1 = 01-01-80 01:00
GMT +2 = 01-01-80 02:00
GMT +3 = 01-01-80 03:00
GMT +4 = 01-01-80 04:00
GMT +5 = 01-01-80 05:00
GMT +6 = 01-01-80 06:00
GMT +7 = 01-01-80 07:00
GMT +8 = 01-01-80 08:00
GMT +9 = 01-01-80 09:00
GMT +10 = 01-01-80 10:00
GMT +11 = 01-01-80 11:00

OFFSET TABLE

GMT +0 = 01-01-80 00:00
GMT +1 = 01-01-80 01:00
GMT +2 = 01-01-80 02:00
GMT +3 = 01-01-80 03:00
GMT +4 = 01-01-80 04:00
GMT +5 = 01-01-80 05:00
GMT +6 = 01-01-80 06:00
GMT +7 = 01-01-80 07:00
GMT +8 = 01-01-80 08:00
GMT +9 = 01-01-80 09:00
GMT +10 = 01-01-80 10:00
GMT +11 = 01-01-80 11:00

GMT +12/-12 = 01-01-80 12:00
GMT -11 = 12-31-80 13:00
GMT -10 = 12-31-80 14:00
GMT -9 = 12-31-80 15:00
GMT -8 = 12-31-80 16:00
GMT -7 = 12-31-80 17:00
GMT -6 = 12-31-80 18:00
GMT -5 = 12-31-80 19:00
GMT -4 = 12-31-80 20:00
GMT -3 = 12-31-80 21:00
GMT -2 = 12-31-80 22:00
GMT -1 = 12-31-80 23:00

@unapibageek - @ssantosv

EVEN BEYOND AAPT

⊞ Return Value

Return a pointer to the structure result, or `NULL` if the date passed to the function is:

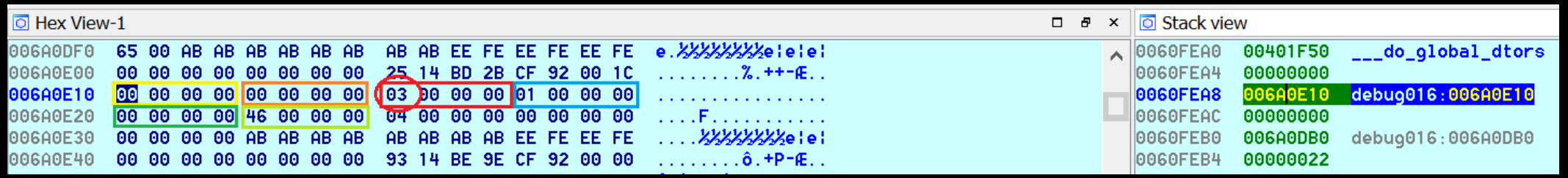
- Before midnight, January 1, 1970.
- After 03:14:07, January 19, 2038, UTC (using `_time32` and `time32_t`).
- After 23:59:59, December 31, 3000, UTC (using `_time64` and `__time64_t`).

EVEN BEYOND AAPT

Return Value

Return a pointer to the structure result, or `NULL` if the date passed to the function is:

- Before midnight, January 1, 1970.
- After 03:14:07, January 19, 2038, UTC (using `_time32` and `time32_t`).
- After 23:59:59, December 31, 3000, UTC (using `_time64` and `__time64_t`).



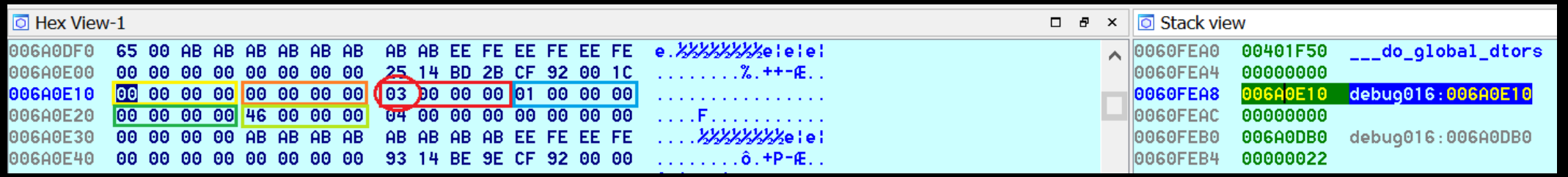
EVEN BEYOND AAPT

Return Value

Return a pointer to the structure result, or `NULL` if the date passed to the function is:

- Before midnight, January 1, 1970.
- After 03:14:07, January 19, 2038, UTC (using `_time32` and `time32_t`).
- After 23:59:59, December 31, 3000, UTC (using `_time64` and `__time64_t`).

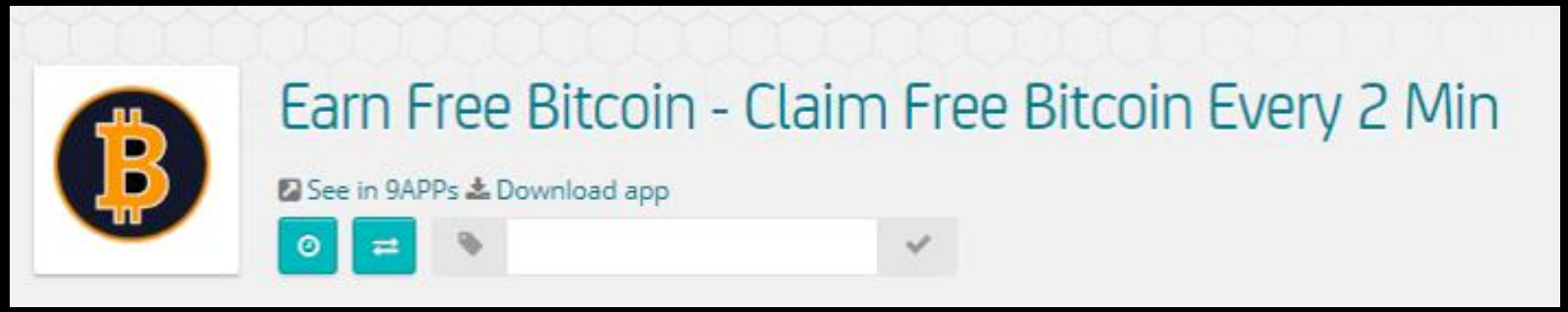
OS X



@unapibageek - @ssantosv

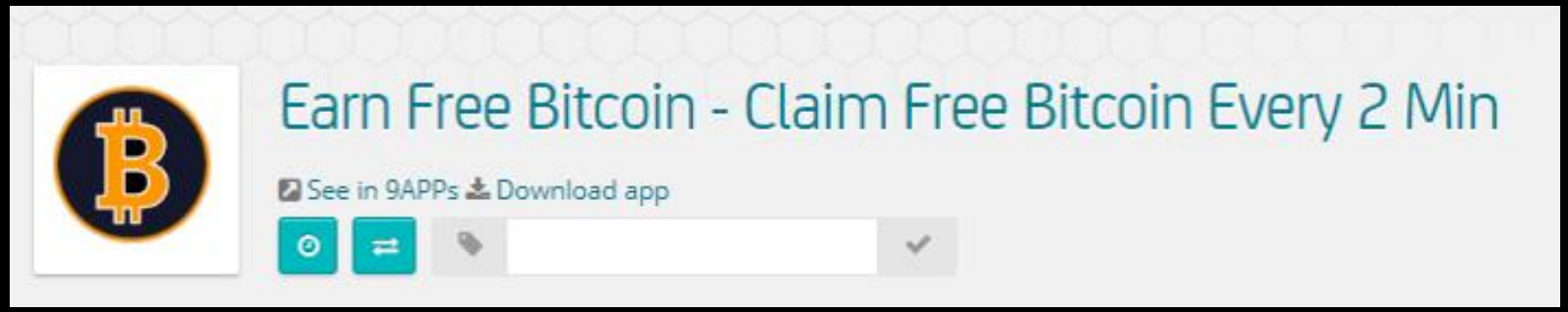

12
1
2
3
4
5
6
7
8
9
10
11

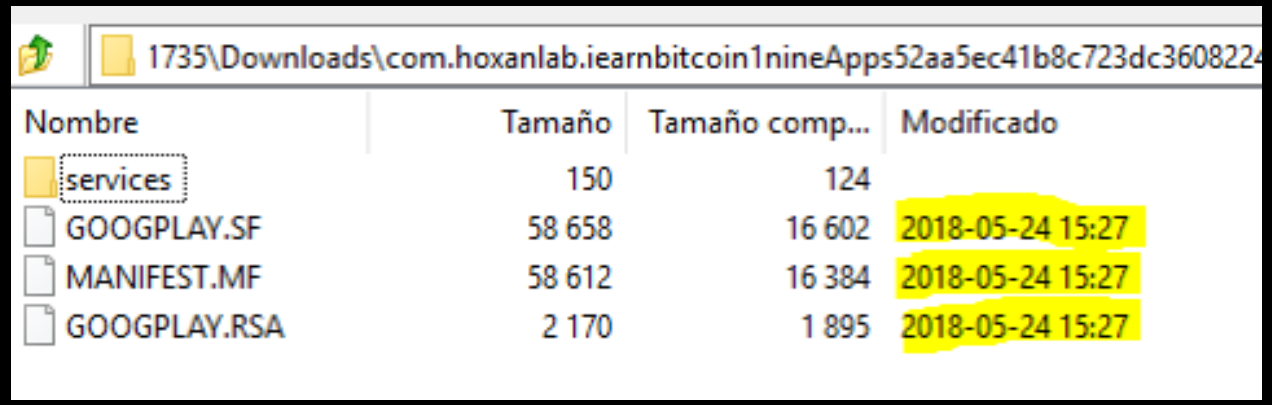
APKS == ZIP



@unapibageek - @ssantosv

APKS == ZIP



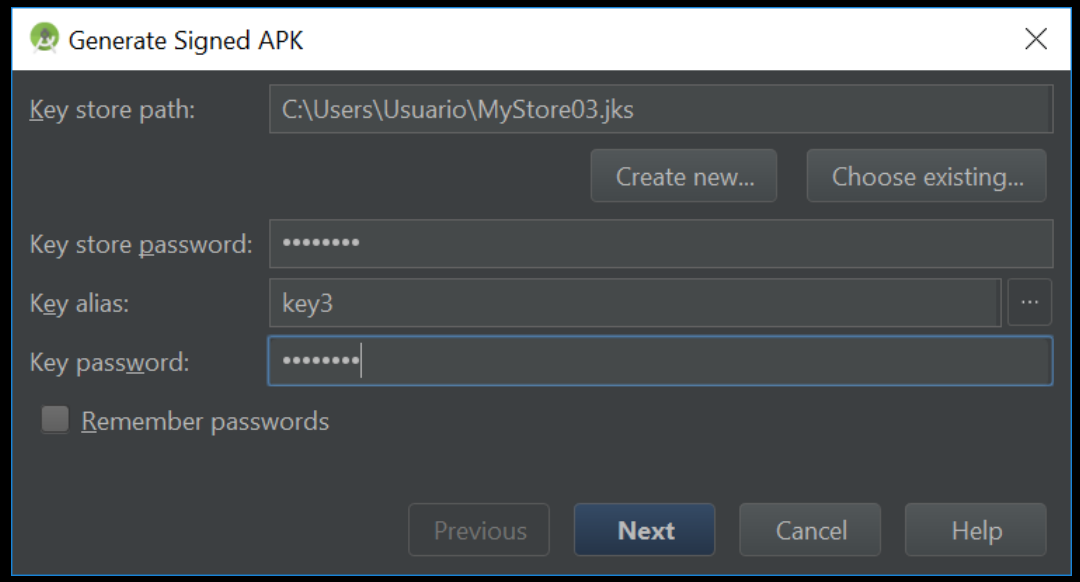


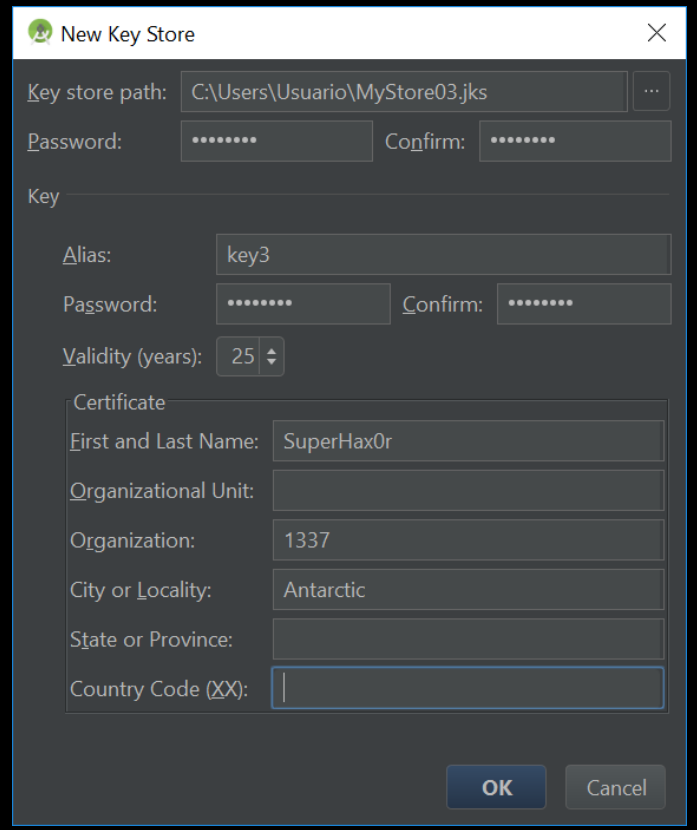
REMINDER:
YYYY-MM-DD

@unapibageek - @ssantosv

SELF SIGNED CERTIFICATES

YOU CAN CREATE CERTIFICATES AD-HOC WHEN YOU ARE ABOUT TO COMPILE YOUR APK

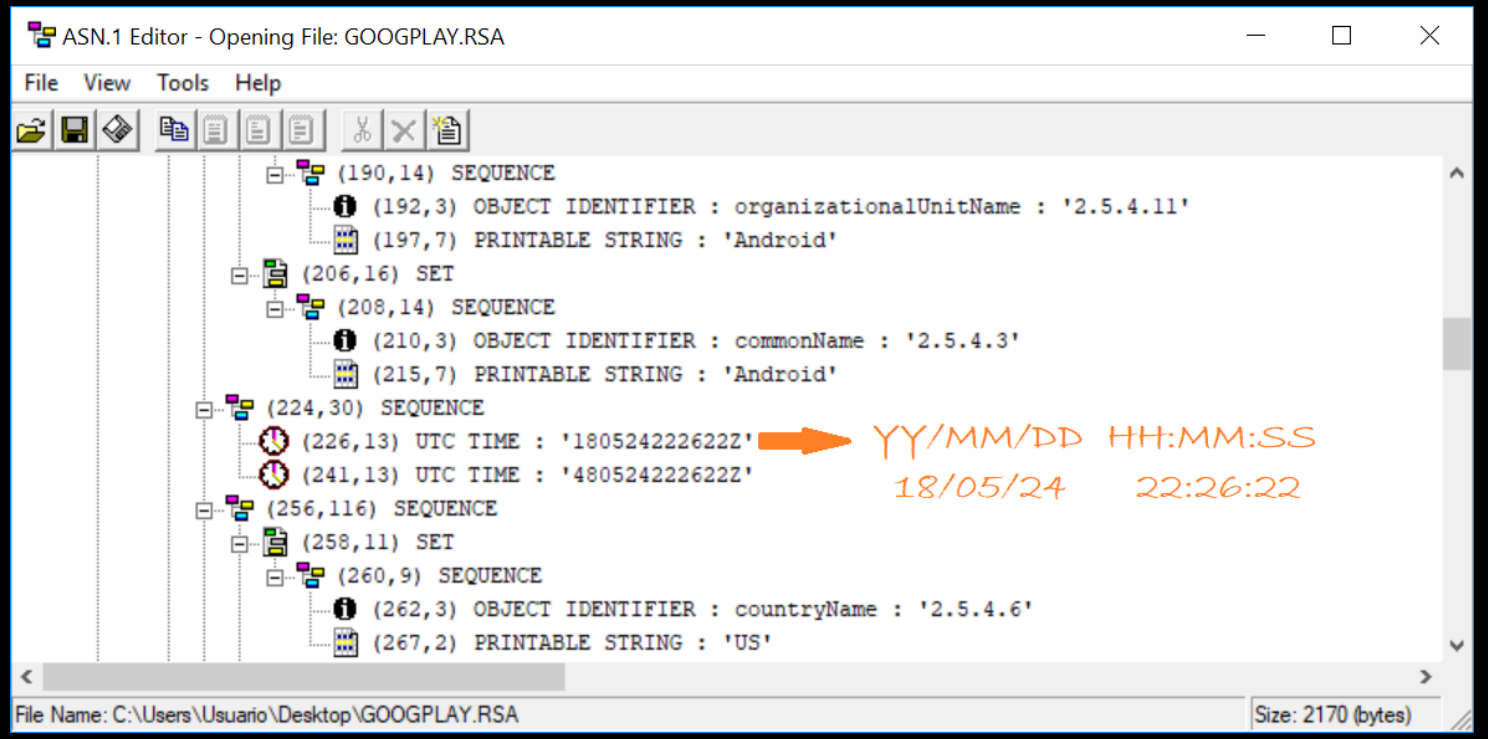




@unapibageek - @ssantosv

SELF SIGNED CERTIFICATES

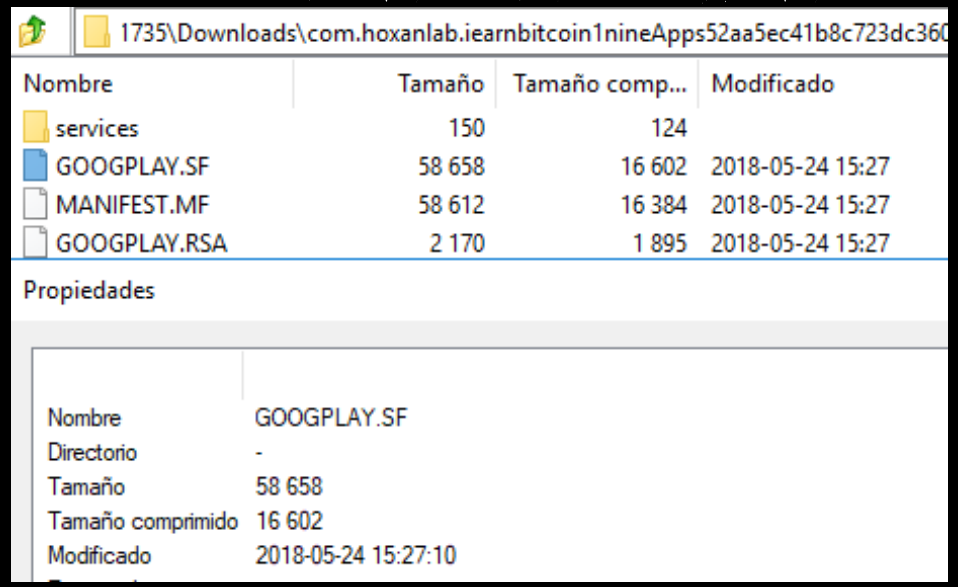
CERTIFICATES STORE THE TIME AND DATE OF THE COMPUTER WHERE THEY HAVE BEEN CREATED, IN UTC TIME



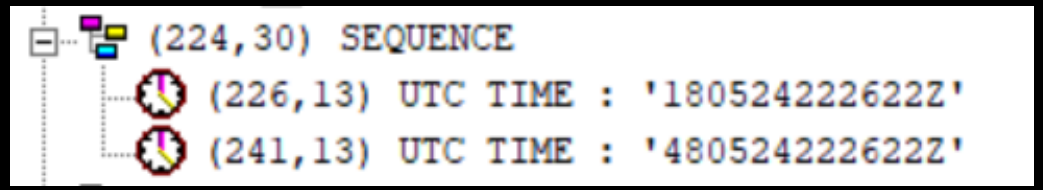
@unapibageek - @ssantosv

DATETIMES

Signature file datetime is the local computer time (timezone included):



Certificate creation datetime (UTC):



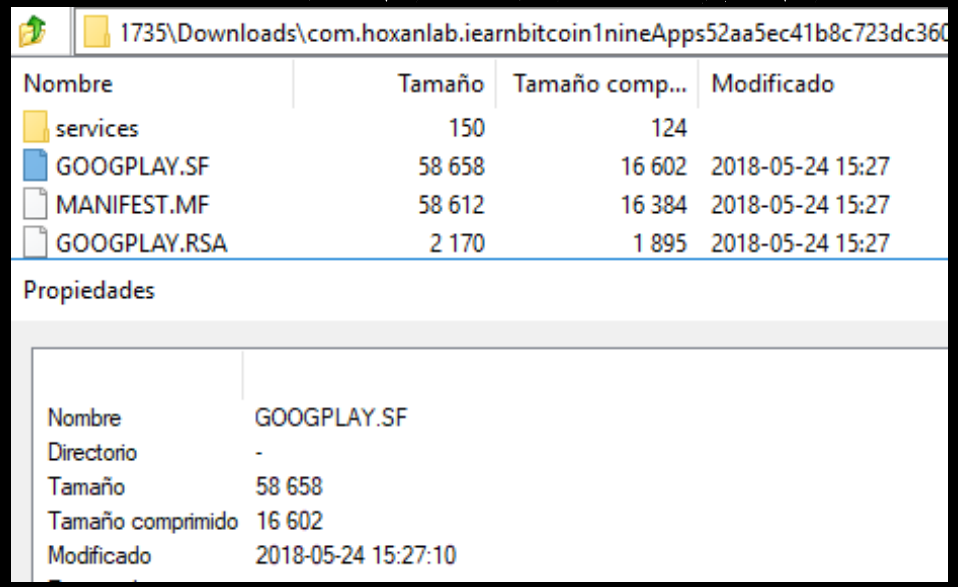
@unapibageek - @ssantosv

DATETIMES

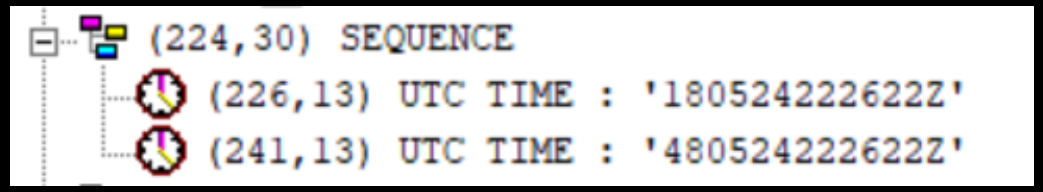
Signature file datetime is the local computer time (timezone included):

+-50 seconds later than the certificate

2018/05/24 15:27:10



Certificate creation datetime (UTC):



2018/05/24 22:26:22

REMINDER: YY-MM-DD

@unapibageek - @ssantosv

ROCK APPROUND THE CLOCK

File time — Cert time

2018/05/24 15:27:10 - 2018/05/24 22:26:22

=

-7 hours and 48 seconds: GMT -7

@unapibageek - @ssantosv

ROCK AROUND THE CLOCK

Another example:

File time | Cert time

2018/07/08 14:30:16 - 2018/07/08 13:30:12

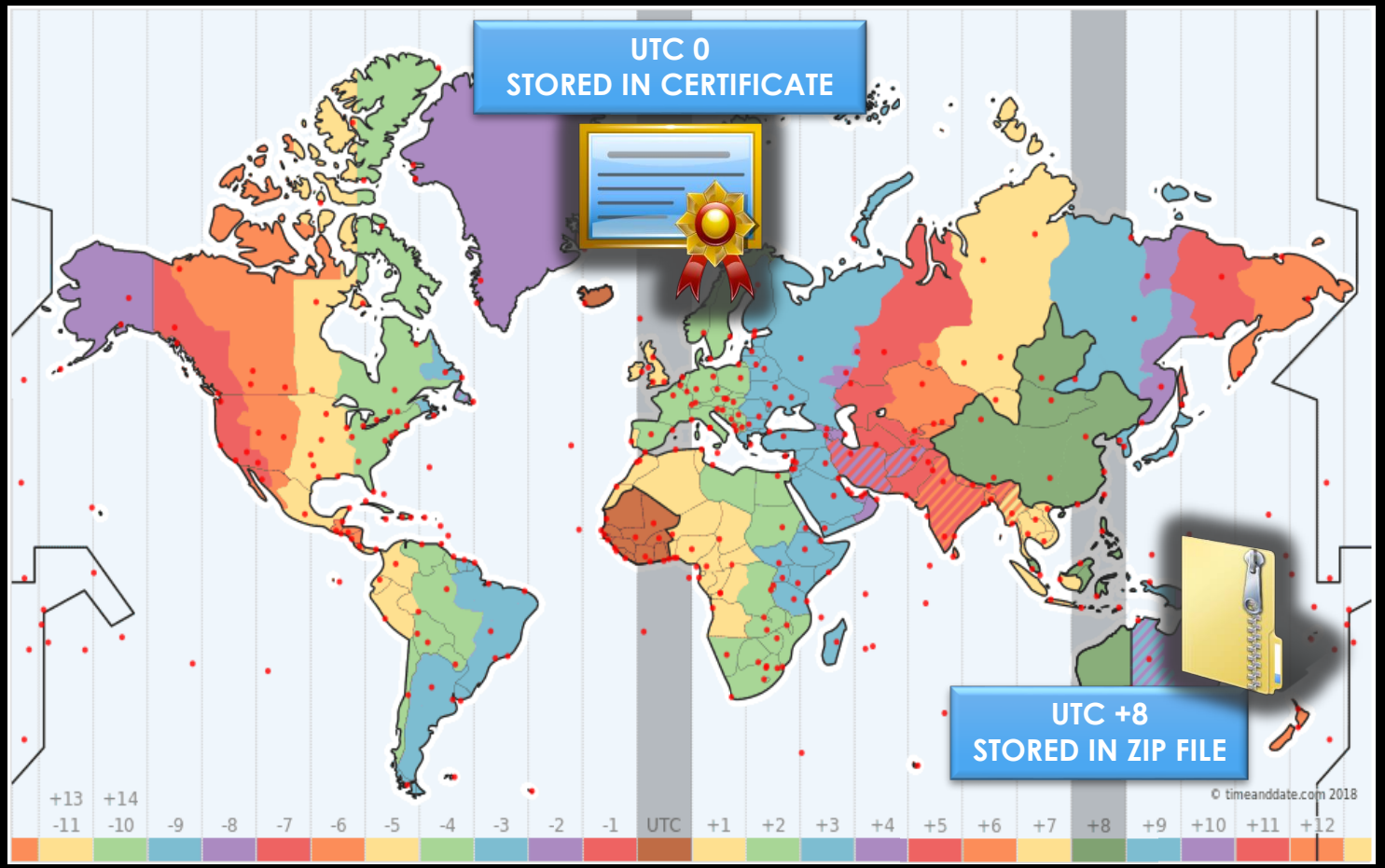
=

1 hour: GMT +1

(File created 4 seconds after the cert...)

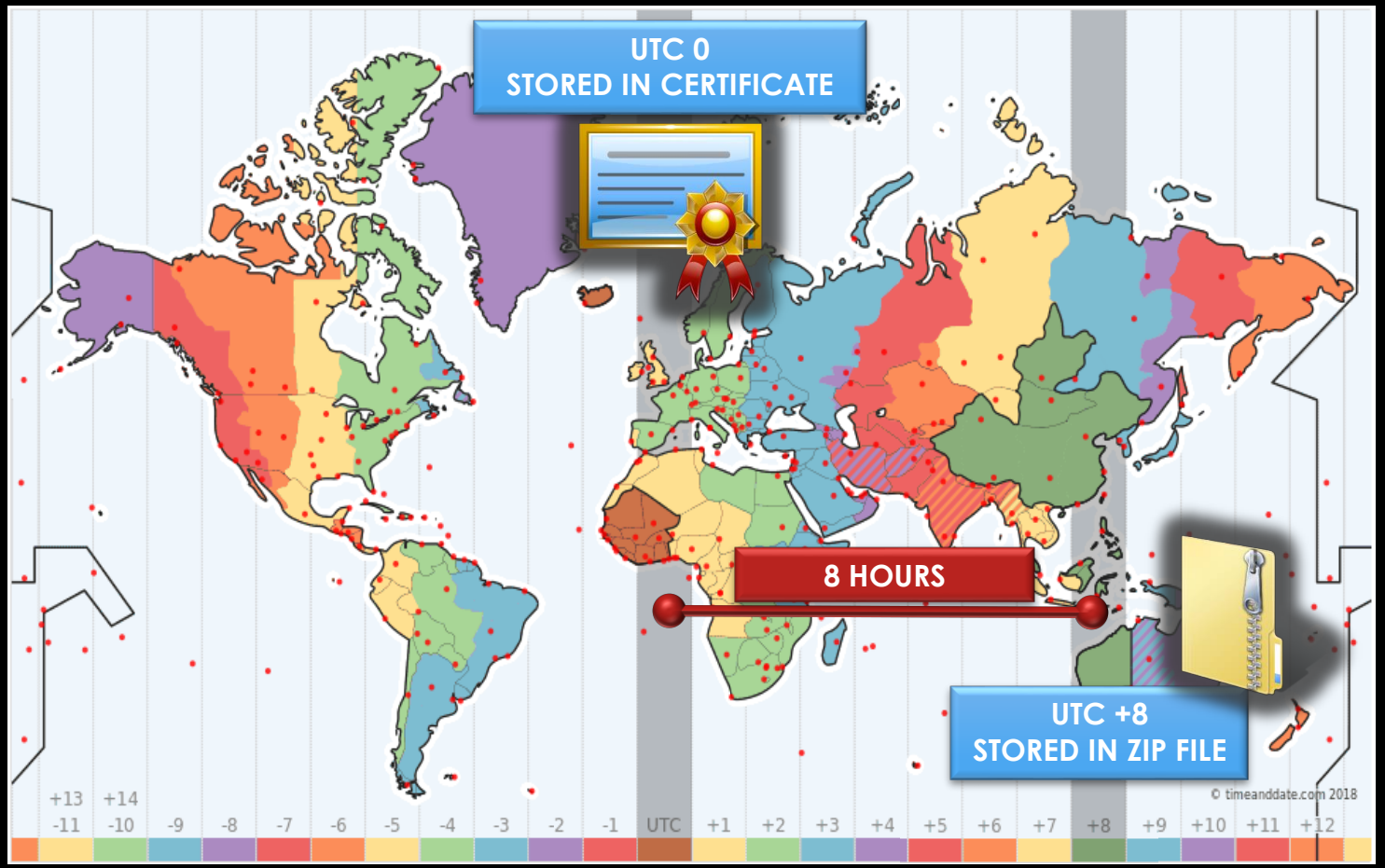
@unapibageek - @ssantosv

MAPPING THE TIMEZONE



Assuming minutes and seconds are "close in time" because certificate and signature are created together

MAPPING THE TIMEZONE



Assuming minutes and seconds are "close in time" because certificate and signature are created together

@unapibageek - @ssantosv

GMT CHECK TOOL

```
D:\f\tools>d:\Python27\python.exe GmtCheck.py "d:\f\adobeupdate.apk"

GmtCheck (c) ElevenPaths 2018. Version: 0.1.0.1

Certificate: /C=RU/ST=State/L=Locality/O=org/OU=org unit/CN=apk/emailAddress=email@fgdf.er

* Certificate creation date (UTC time): 2016-12-30 08:11:17
* Signature file date seems to be (system time): 2016-12-30 11:11:18

If we assume they were created at the same moment, the developer's time zone is: UTC + 3 (Baghdad, K
uwait, Nairobi, Riyadh)

Signature file was created 0 minutes 1 seconds after the certificate. So results seem accurate.

D:\f\tools>
```

@unapibageek - @ssantosv

GMT CHECK TOOL

```
D:\f\tools>d:\Python27\python.exe GmtCheck.py "d:\f\adobeupdate.apk"

GmtCheck (c) ElevenPaths 2018. Version: 0.1.0.1

Certificate: /C=RU/ST=State/L=Locality/O=org/OU=org unit/CN=apk/emailAddress=email@fgdf.er

* Certificate creation date (UTC time): 2016-12-30 08:11:17
* Signature file date seems to be (system time): 2016-12-30 11:11:18

If we assume they were created at the same moment, the developer's time zone is: UTC + 3 (Baghdad, K
uwait, Nairobi, Riyadh)

Signature file was created 0 minutes 1 seconds after the certificate. So results seem accurate.

D:\f\tools>
```

@unapibageek - @ssantosv

GMT CHECK TOOL

```
D:\f\tools>d:\Python27\python.exe GmtCheck.py "d:\f\adobeupdate.apk"

GmtCheck (c) ElevenPaths 2018. Version: 0.1.0.1

Certificate: /C=RU/ST=State/L=Locality/O=org/OU=org unit/CN=apk/emailAddress=email@fgdf.er

* Certificate creation date (UTC time): 2016-12-30 08:11:17
* Signature file date seems to be (system time): 2016-12-30 11:11:18

If we assume they were created at the same moment, the developer's time zone is: UTC + 3 (Baghdad, K
uwait, Nairobi, Riyadh)

Signature file was created 0 minutes 1 seconds after the certificate. So results seem accurate.

D:\f\tools>
```

@unapibageek - @ssantosv

GMT CHECK TOOL

```
D:\f\tools>d:\Python27\python.exe GmtCheck.py "d:\f\adobeupdate.apk"

GmtCheck (c) ElevenPaths 2018. Version: 0.1.0.1

Certificate: /C=RU/ST=State/L=Locality/O=org/OU=org unit/CN=apk/emailAddress=email@fgdf.er

* Certificate creation date (UTC time): 2016-12-30 08:11:17
* Signature file date seems to be (system time): 2016-12-30 11:11:18

If we assume they were created at the same moment, the developer's time zone is: UTC + 3 (Baghdad, K
uwait, Nairobi, Riyadh)

Signature file was created 0 minutes 1 seconds after the certificate. So results seem accurate.

D:\f\tools>
```

@unapibageek - @ssantosv

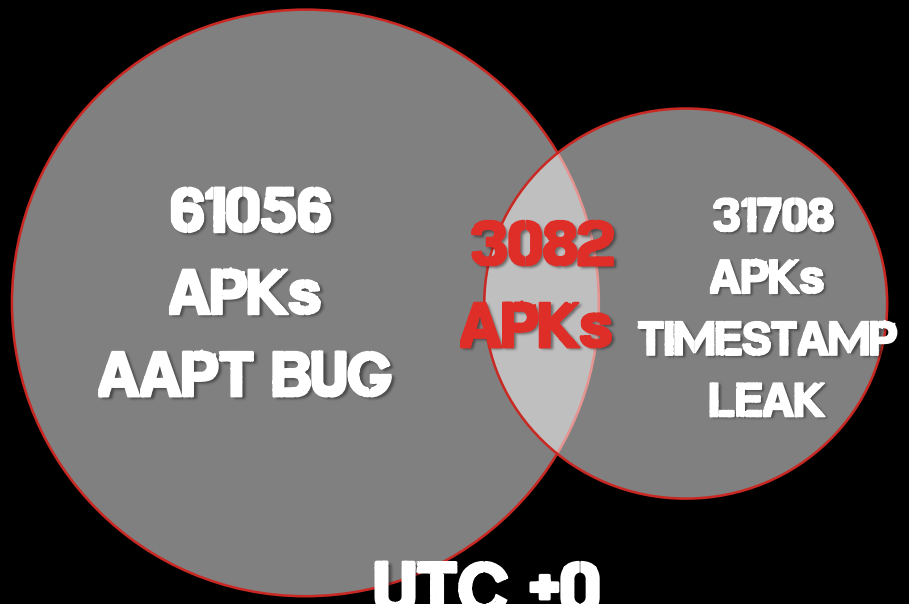
STATISTICS

@unapibageek - @ssantosv

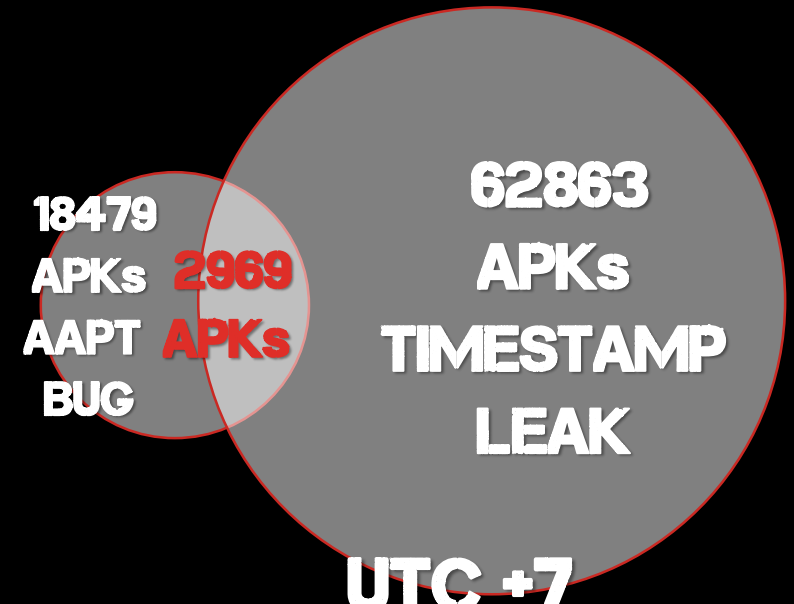
STATISTICS

TIMEZONE LEAKAGE BY AAPT BUG: 179.122 APKs

TIMEZONE LEAKAGE BY DATETIMES: 477.849 APKs



UTC +0



UTC +7

MORE THAN HALF A MILLION OF APKs LEAKING THEIR TIMEZONE FROM OUR 10 MILLION APKS DATABASE

@unapibageek - @ssantosv

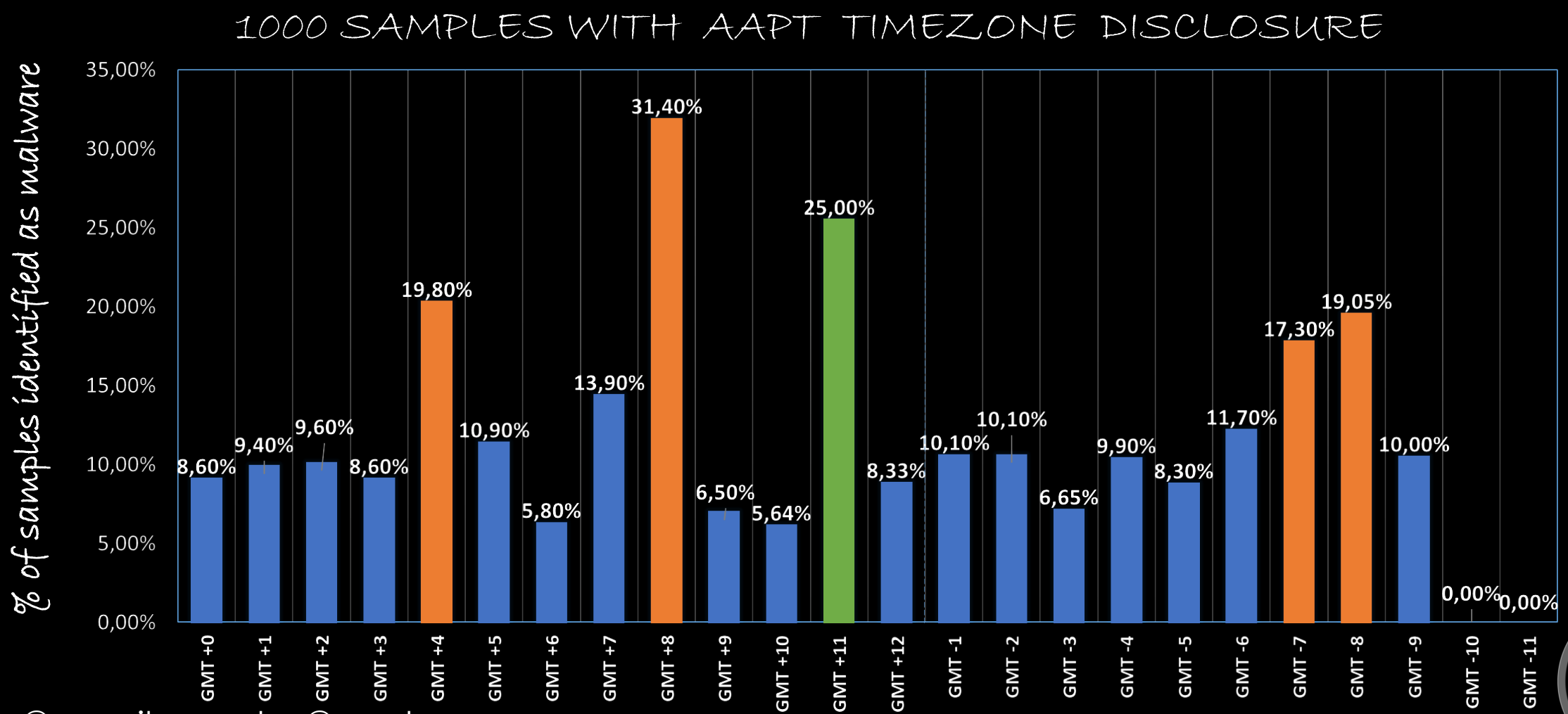
IS THIS USEFUL FOR MALWARE?

MALWARE (1000 SAMPLES) ANALYZED WITH AAPT TIMEZONE DISCLOSURE

Type	# of APKs	Detected only by 1 AV	Detected only by 2 AV	Detected only by 3 AV	Detected by +3 AV	TOTAL	% detected
UTC +0	1000	45	13	6	22	86	8,60%
UTC +1	1000	55	5	4	30	94	9,40%
UTC +2	1000	60	6	4	26	96	9,60%
UTC +3	1000	38	21	6	21	86	8,60%
UTC +4	1000	71	28	27	72	198	19,80%
UTC +5	1000	74	7	6	22	109	10,90%
UTC +6	1000	54	0	1	3	58	5,80%
UTC +7	1000	66	18	9	46	139	13,90%
UTC +8	1000	102	47	39	126	314	31,40%
UTC +9	1000	57	4	0	4	65	6,50%
UTC +10	532	15	3	2	10	30	5,64%
UTC +11	276	18	0	4	47	69	25,00%
UTC +12	72	6	0	0	0	6	8,33%
UTC -1	1000	61	10	11	19	101	10,10%
UTC -2	1000	42	25	17	17	101	10,10%
UTC -3	391	19	2	3	2	26	6,65%
UTC -4	1000	74	3	6	16	99	9,90%
UTC -5	1000	53	13	11	6	83	8,30%
UTC -6	1000	30	10	9	68	117	11,70%
UTC -7	1000	92	11	8	62	173	17,30%
UTC -8	21	3	1	0	0	4	19,05%
UTC -9	10	0	1	0	0	1	10,00%
UTC -10	2	0	0	0	0	0	0,00%
UTC -11	6	0	0	0	0	0	0,00%

@unapibageek - @ssantosv

STATISTICS



@unapibageek - @ssantosv

IS THIS USEFUL FOR MALWARE?

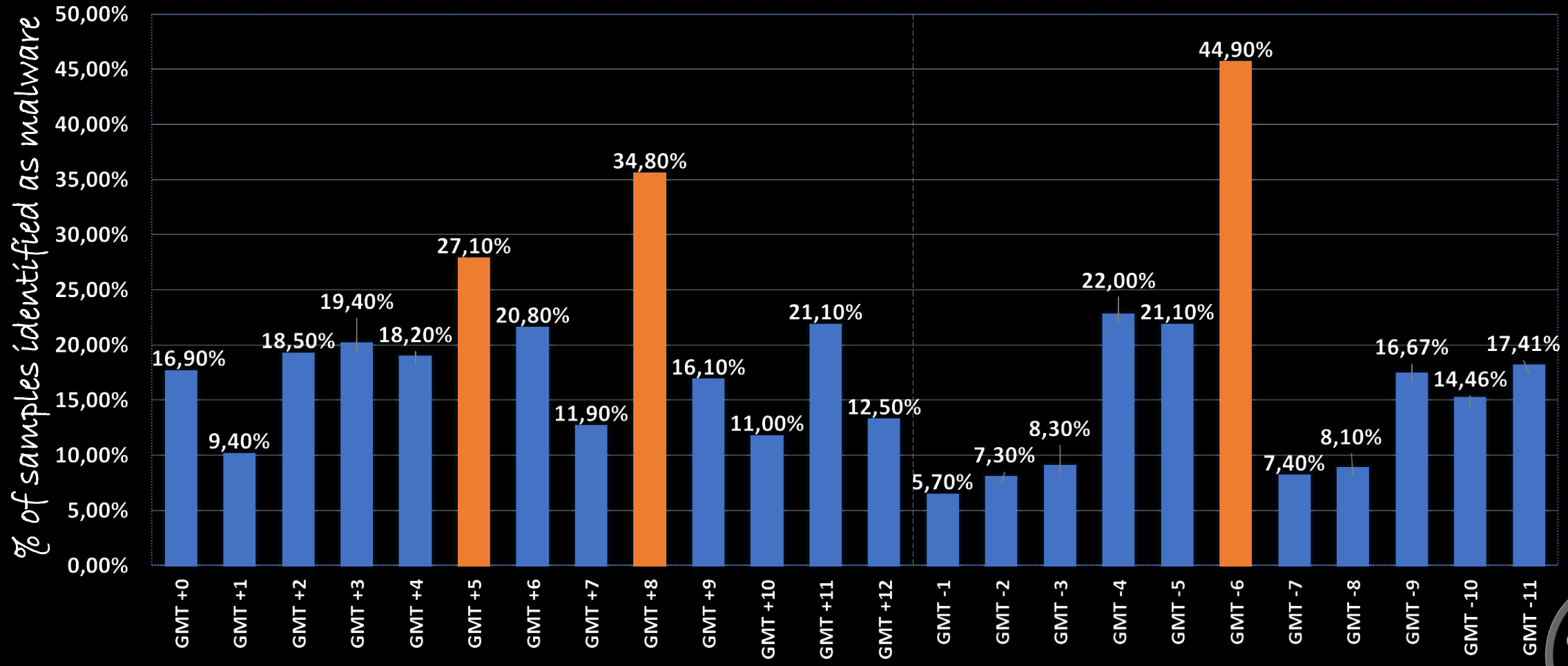
MALWARE (1000 SAMPLES) ANALYZED WITH FILE/CERTIFICATE DATETIMES

Type	# of APKs	Detected only by 1 AV	Detected only by 2 AV	Detected only by 3 AV	Detected by +3 AV	TOTAL	% detected
UTC +0	1000	35	16	11	107	169	16,90%
UTC +1	1000	58	8	2	26	94	9,40%
UTC +2	1000	76	13	16	80	185	18,50%
UTC +3	1000	91	28	15	60	194	19,40%
UTC +4	1000	72	27	18	65	182	18,20%
UTC +5	1000	58	29	31	153	271	27,10%
UTC +6	1000	98	24	16	70	208	20,80%
UTC +7	1000	53	20	3	43	119	11,90%
UTC +8	1000	83	31	16	218	348	34,80%
UTC +9	1000	71	51	3	36	161	16,10%
UTC +10	1000	43	7	16	44	110	11,00%
UTC +11	1000	53	12	7	139	211	21,10%
UTC +12	1000	55	10	1	59	125	12,50%
UTC -1	1000	17	6	3	31	57	5,70%
UTC -2	1000	29	9	6	29	73	7,30%
UTC -3	1000	36	3	4	40	83	8,30%
UTC -4	1000	59	11	5	145	220	22,00%
UTC -5	1000	49	7	2	153	211	21,10%
UTC -6	1000	43	15	8	383	449	44,90%
UTC -7	1000	33	11	7	23	74	7,40%
UTC -8	1000	35	10	4	32	81	8,10%
UTC -9	276	16	4	2	24	46	16,67%
UTC -10	588	36	21	4	24	85	14,46%
UTC -11	293	26	4	1	20	51	17,41%

@unapibageek - @ssantosv

STATISTICS

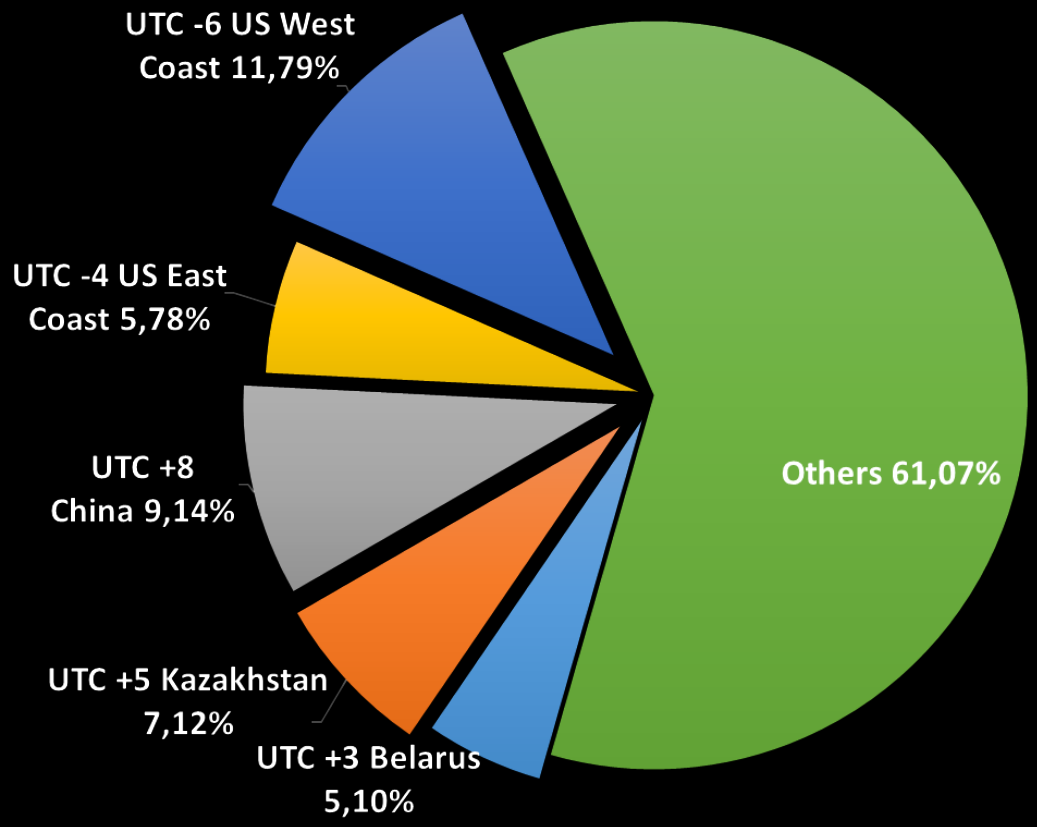
1000 APKS WITH FILE/CERTIFICATE DATETIMES (do not forget DST!)



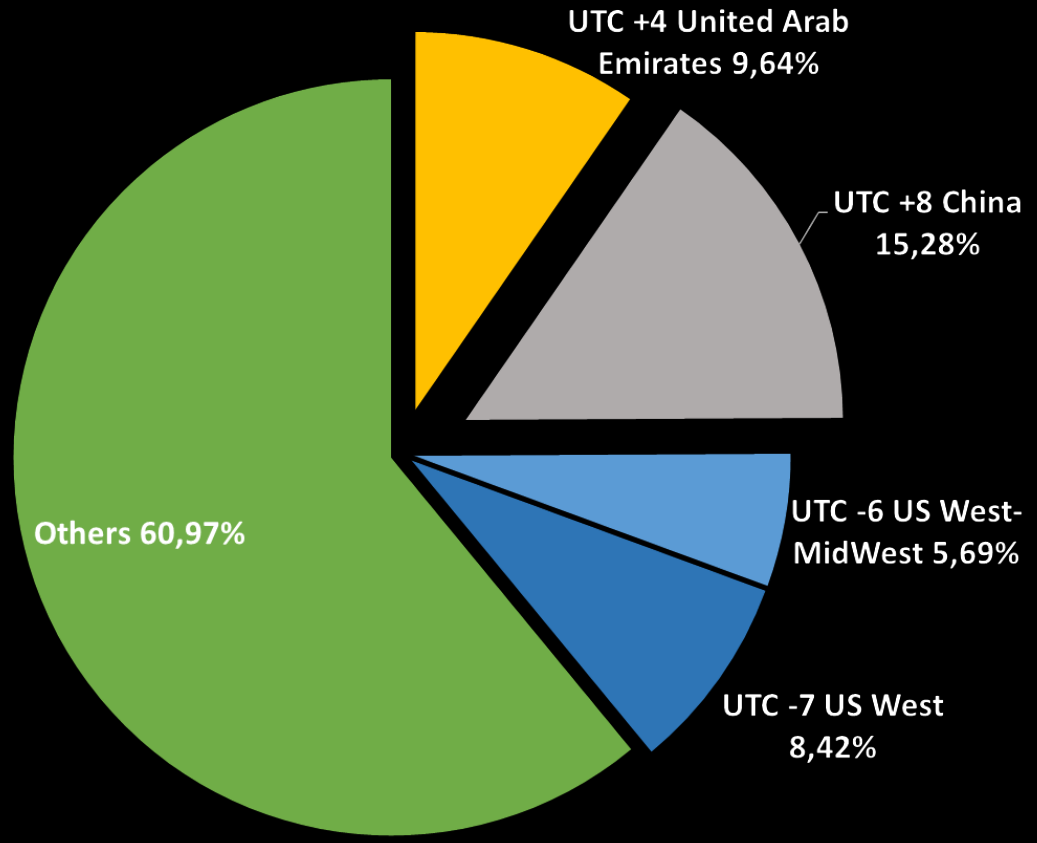
@unapibageek - @ssantosv

STATISTICS

DISTRIBUTION OF MALWARE BY TIMEZONE AND USING...



3807 APKS IDENTIFIED AS MALWARE WITH FILE/CERTIFICATE DATETIMES



2055 APKS IDENTIFIED AS MALWARE WITH AAPT TIMEZONE DISCLOSURE

@unapibageek - @ssantosv

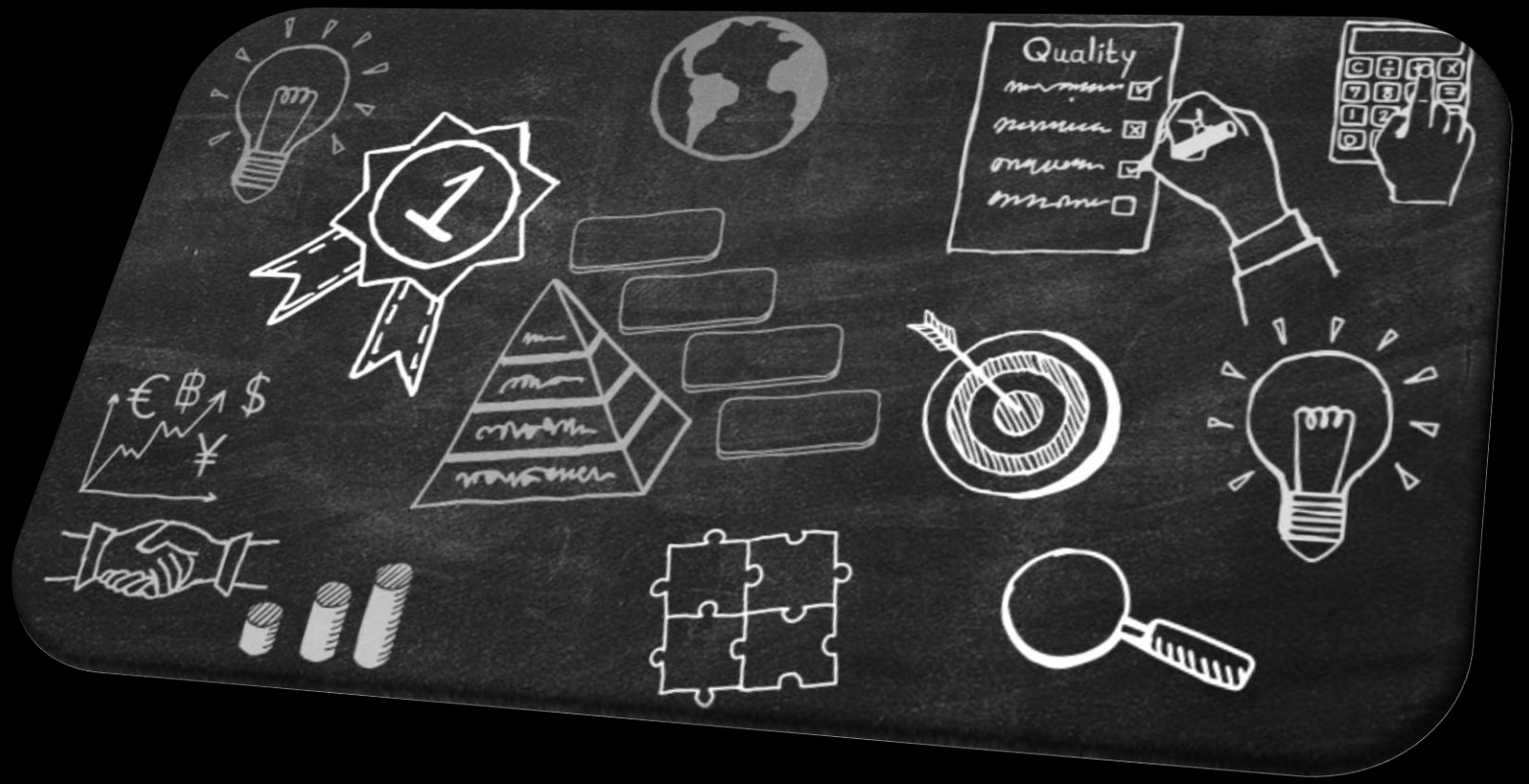
STATISTICS

IN OUR DATABASE, THE STANDARD RATE IS: FOR EACH RANDOM 1000APKS WE IDENTIFY 60 AS MALWARE, SO:

1000 APKs sample	# APKs Detected by +3 AV	Times more likely to be malware than Standard Rate
UTC +5	153	2,55
UTC +8	218	3,63
UTC -5	153	2,55
UTC -6	383	6,38
UTC +4	72	1,20
UTC +8	126	2,10

IN SHORT: AN UTC-6 APK IS 6,38 TIMES MORE LIKELY TO BE MALWARE THAN STANDARD RATE

EXAMPLES



EXAMPLES

DeathRing: Pre-loaded malware hits smartphones for the second time in 2014

By Jeremy Linden

When you walk out of a retailer with a shiny new phone, you trust that it's clean and safe to use. But this might not always be the case, as evidenced by the latest pre-loaded malware Lookout identified called DeathRing. DeathRing is a Chinese Trojan that is pre-installed on a number of smartphones most popular in Asian and African countries. Detection volumes are moderate, though we consider this a concerning threat given its pre-loaded nature and the fact that we are actively seeing detections of it around the world.

What does it do?

The Trojan masquerades as a ringtone app, but instead can download SMS and WAP content from its command and control server to the victim's phone. It can then use this content for malicious means. For example, DeathRing might use SMS content to phish victim's personal information by fake text messages requesting the desired data. It may also use WAP, or browser, content to prompt victims to download further APKs -- concerning given that the malware authors could be tricking people into downloading further malware that

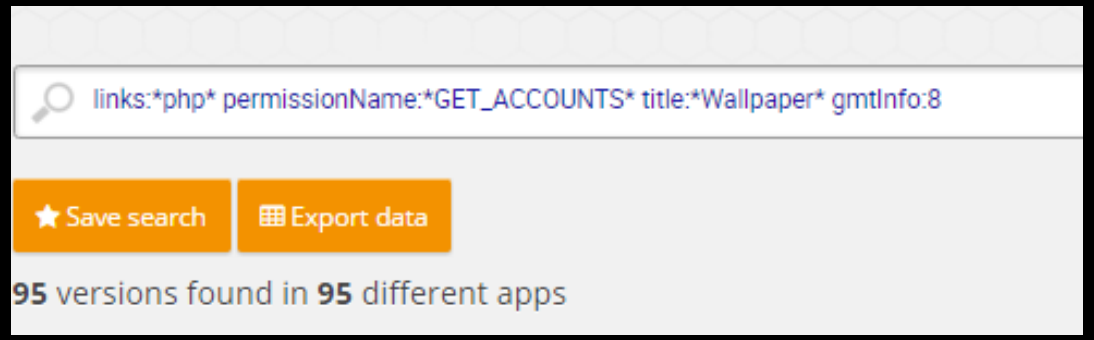
FILE/CERTIFICATE DATETIMES
REAL EXAMPLE: **DEATHRING**

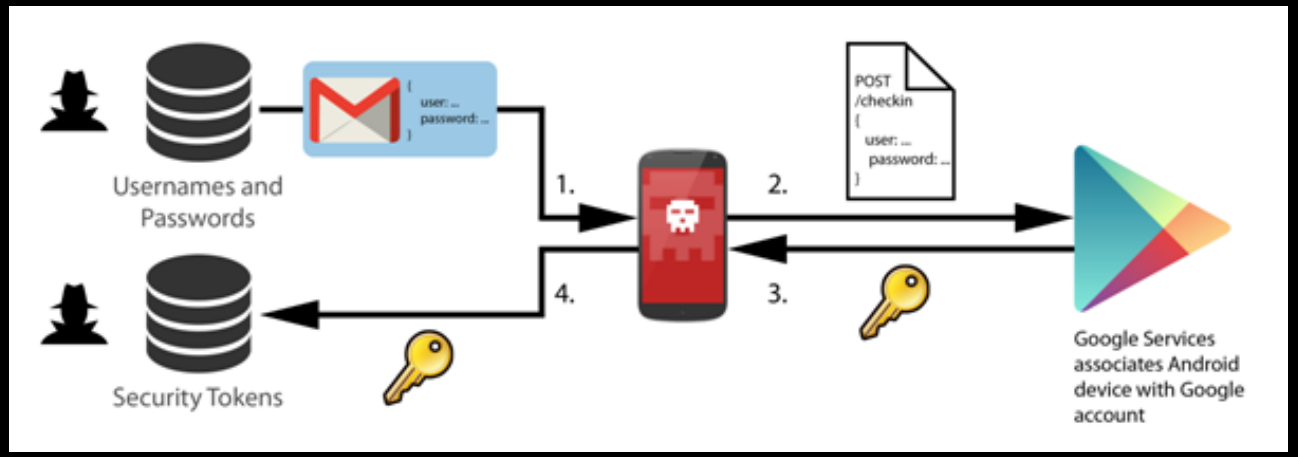
AAPT TIMEZONE DISCLOSURE REAL EXAMPLE: **JUDY**

@unapibageek - @ssantosv

EXAMPLES

CHINESE COMPILED APKS IN THE SPOTLIGHT





SC MEDIA
The Cyber-Security source
HOME | NEWS & FEATURES | BUYER'S GUIDE |
SC UK | SC US

Top of the app charts - Shuabang: automated malware made in China

Jul 27, 2015

@unapibageek - @ssantosv

EXAMPLES

HIDDAD MALWARE, COMES FROM...

Hiddad Android malware gets top user ratings for all the wrong reasons

Mihai Grigorescu May 4, 2018 Android, Security & Privacy 0 Comments

Most malware keeps a low profile. But apps containing Hiddad, an ad-distributing malware for Androids, get lots of 5-star ratings from infected users – and not for the usual reasons. Hiddad is presented as a YouTube downloading app and usually goes to market on Google Play labeled as Tube mate or Snap Tube.

EXAMPLES

BOTH TECHNIQUES EXAMPLES:

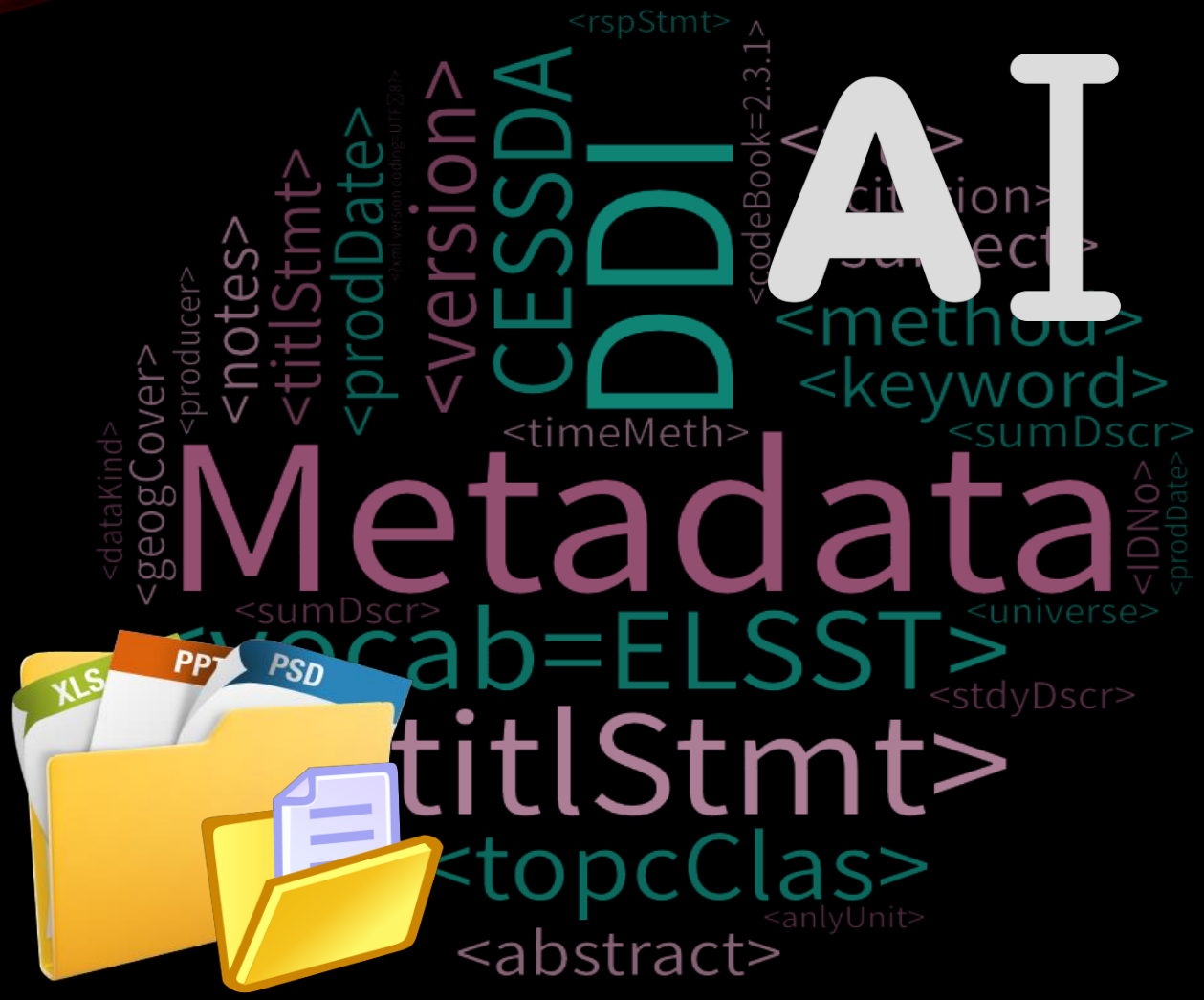
certificateValidFrom	signatureFileLastUpdateDate	oldestDateFile	File/Certificate DateTime Technique
2016/12/04 07:27:57	2016/12/04 10:30:14	1980/01/01 03:00:00	03:02:17
2016/12/03 16:21:02	2016/12/03 19:24:30	1980/01/01 03:00:00	03:03:28
2016/12/03 11:52:30	2016/12/03 14:55:54	1980/01/01 03:00:00	03:03:24

EXAMPLES

BOTH TECHNIQUES EXAMPLES:

certificateValidFrom	signatureFileLastUpdateDate	oldestDateFile	File/Certificate DateTime Technique
2016/12/04 07:27:57	2016/12/04 10:30:14	1980/01/01 03:00:00	03:02:17
2016/12/03 16:21:02	2016/12/03 19:24:30	1980/01/01 03:00:00	03:03:28
2016/12/03 11:52:30	2016/12/03 14:55:54	1980/01/01 03:00:00	03:03:24

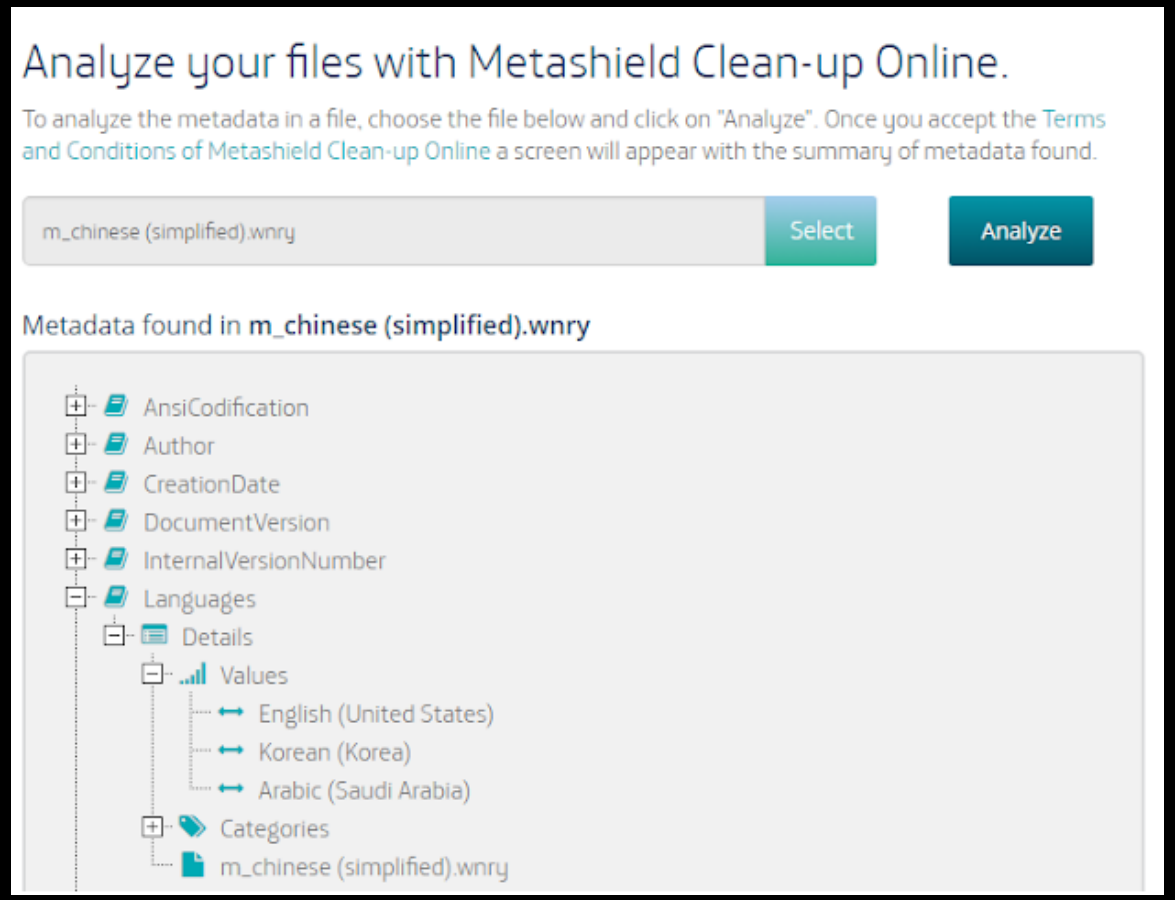
FULLY AUTOMATED?



@unapibageek - @ssantosv

METADATA

WANNACRY METADATA, THE INSPIRATION



All the files, no matter the language they are written in, uses **English, Korean and Arabic** as metadata "Language". These are the language that Word expects you to write in, and are filtered trhough Word to RTF files. Why this three? Does he uses all of them? No.

- Arabic is present if you have a specific version of Word (\deflangfe is the keyword for "Default language ID for Asian versions of Word".) It does not mean this language is configured as predeterminded in Word for corrections. It means you use a specific kind of EMEA version of Word, very common. You can tell if your RTF files have \adeflang1025 in them.
- Korean is the language in \deflang which defines the default language used in the document used with a \plain control word. **That means, he has Korean and English as default "usual" languages in Word**
- English is present because of how Word works. If you set your Word with Korean as deafult language, it will always be set with English as well. Seems like a Word feature.

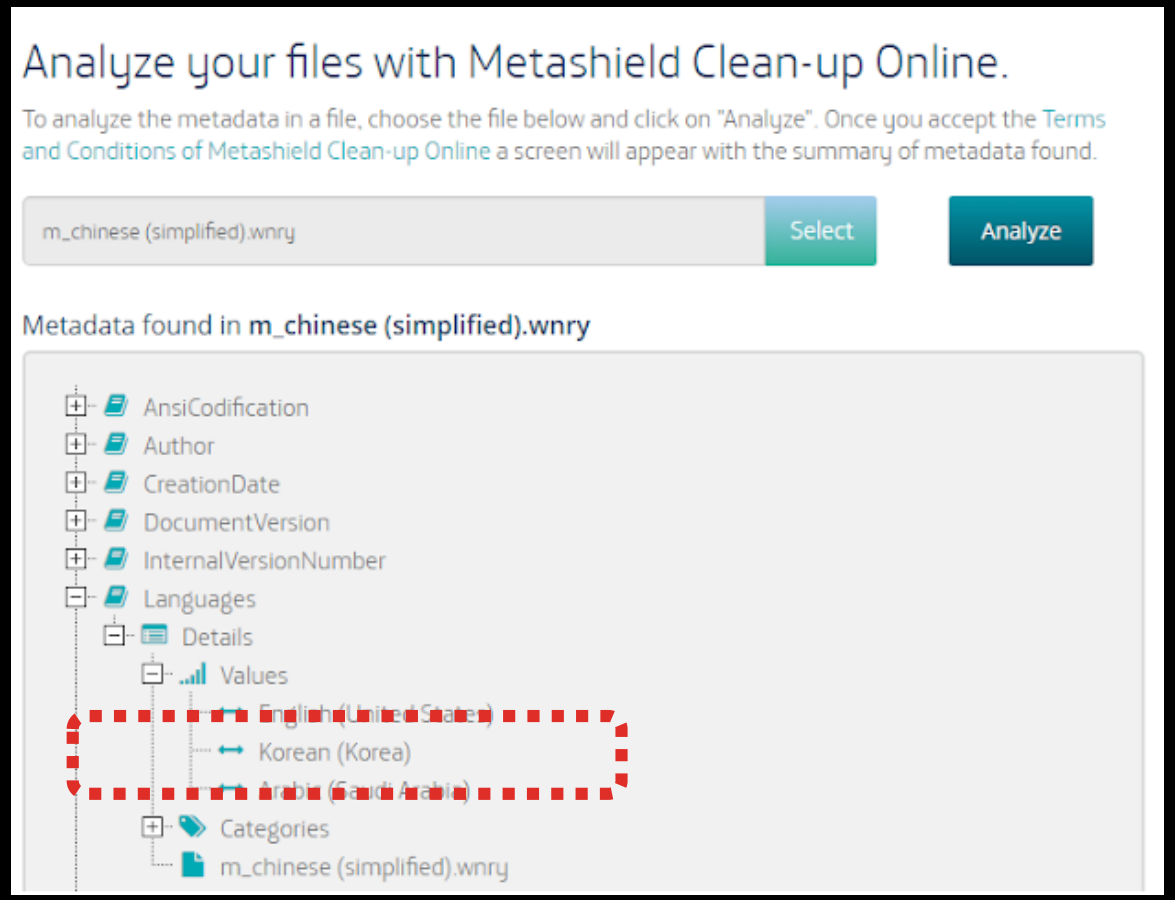
This is the raw data "inside" the RTF.

```
\rtf1\adeflang1025\ansi\ansicpg1252\uc2\adeff31507\deff0\stshfdbch31505\stshfloch31
506\stshfhich31506\stshfbi0\deflang1033\deflangfe1042\
```

@unapibageek - @ssantosv

METADATA

WANNACRY METADATA, THE INSPIRATION



All the files, no matter the language they are written in, uses **English, Korean and Arabic** as metadata "Language". These are the language that Word expects you to write in, and are filtered trhough Word to RTF files. Why this three? Does he uses all of them? No.

- Arabic is present if you have a specific version of Word (\deflangfe is the keyword for "Default language ID for Asian versions of Word".) It does not mean this language is configured as predeterminded in Word for corrections. It means you use a specific kind of EMEA version of Word, very common. You can tell if your RTF files have \adeflang1025 in them.
- Korean is the language in \deflang which defines the default language used in the document used with a \plain control word. **That means, he has Korean and English as default "usual" languages in Word**
- English is present because of how Word works. If you set your Word with Korean as deafult language, it will always be set with English as well. Seems like a Word feature.

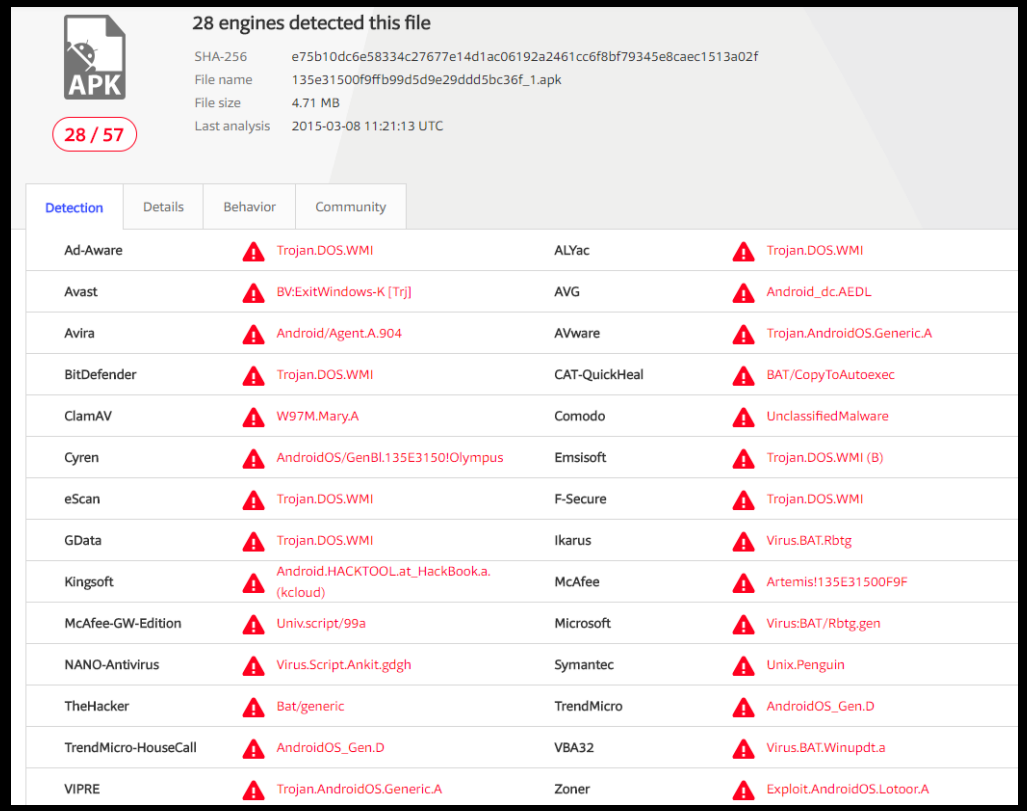
This is the raw data "inside" the RTF.

```
\rtf1\adeflang1025\ansi\ansicpg1252\uc2\adeff31507\deff0\stshfdbch31505\stshfloch31
506\stshfhich31506\stshfbi0\deflang1033\deflangfe1042\
```

@unapibageek - @ssantosv

METADATA

WAS IT USEFUL IN ANDROID MALWARE?



28 engines detected this file

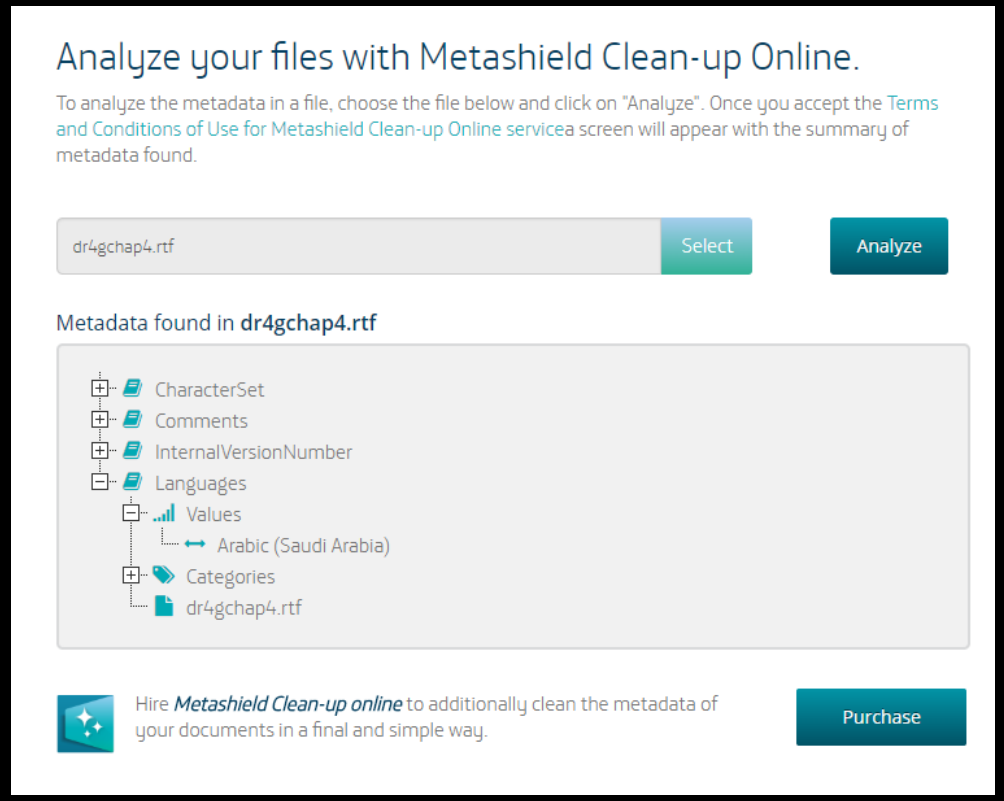
APK

28 / 57

SHA-256	e75b10dc6e58334c27677e14d1ac06192a2461cc6f8bf79345e8caec1513a02f
File name	135e31500f9ffb99d5d9e29ddd5bc36f_1.apk
File size	4.71 MB
Last analysis	2015-03-08 11:21:13 UTC

Detection | Details | Behavior | Community

Ad-Aware	Trojan.DOS.WMI	ALYac	Trojan.DOS.WMI
Avast	BV:ExitWindows-K [Trj]	AVG	Android_dc.AEDL
Avira	Android/Agent.A.904	AVware	Trojan.AndroidOS.Generic.A
BitDefender	Trojan.DOS.WMI	CAT-QuickHeal	BAT/CopyToAutoexec
ClamAV	W97M.Mary.A	Comodo	UnclassifiedMalware
Cyren	AndroidOS/GenBl.135E3150!Olympus	Emsisoft	Trojan.DOS.WMI (B)
eScan	Trojan.DOS.WMI	F-Secure	Trojan.DOS.WMI
GData	Trojan.DOS.WMI	Ikarus	Virus.BAT.Rbtg
Kingsoft	Android.HACKTOOL.at_HackBook.a. (kcloud)	McAfee	Artemis!135E31500F9F
McAfee-GW-Edition	Univ.script/99a	Microsoft	Virus:BAT/Rbtg.gen
NANO-Antivirus	Virus.Script.Ankit.gdgh	Symantec	Unix.Penguin
TheHacker	Bat/generic	TrendMicro	AndroidOS_Gen.D
TrendMicro-HouseCall	AndroidOS_Gen.D	VBA32	Virus.BAT.Winupdt.a
VIPRE	Trojan.AndroidOS.Generic.A	Zoner	Exploit.AndroidOS.Lotoor.A



Analyze your files with Metashield Clean-up Online.

To analyze the metadata in a file, choose the file below and click on "Analyze". Once you accept the Terms and Conditions of Use for Metashield Clean-up Online servicea screen will appear with the summary of metadata found.

dr4gchap4.rtf Select Analyze

Metadata found in **dr4gchap4.rtf**

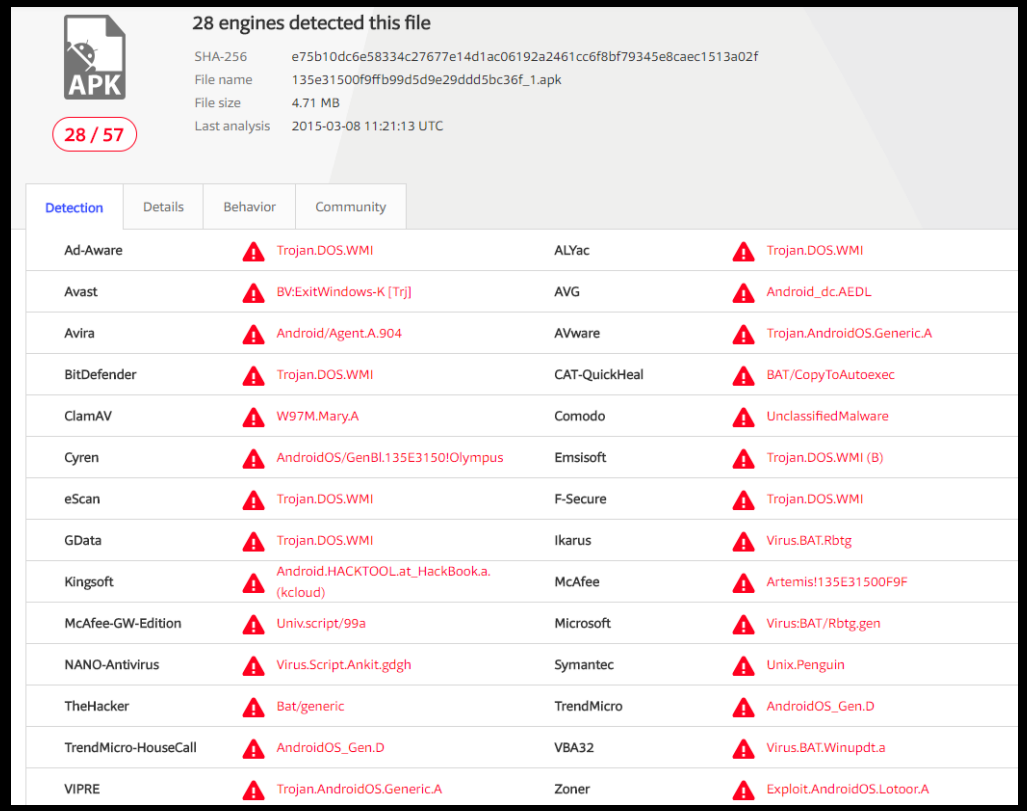
- CharacterSet
- Comments
- InternalVersionNumber
- Languages
  - Values
    - Arabic (Saudi Arabia)
  - Categories
  - dr4gchap4.rtf

Hire *Metashield Clean-up online* to additionally clean the metadata of your documents in a final and simple way. Purchase

@unapibageek - @ssantosv

METADATA

WAS IT USEFUL IN ANDROID MALWARE?



28 engines detected this file

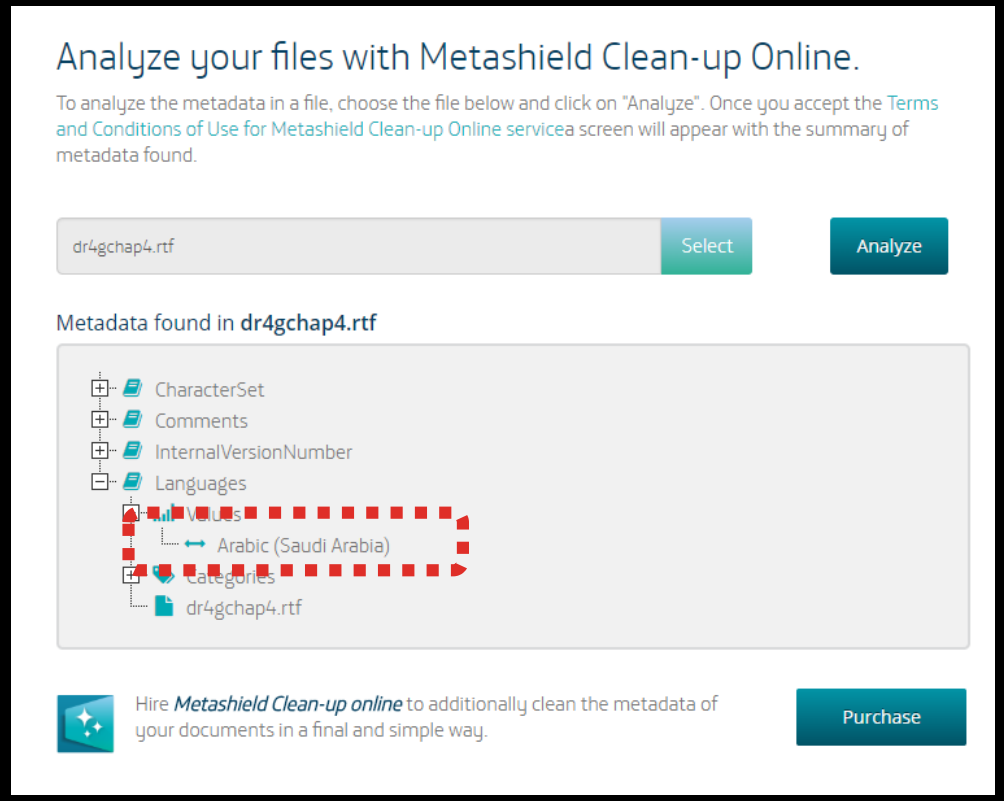
APK

28 / 57

SHA-256	e75b10dc6e58334c27677e14d1ac06192a2461cc6f8bf79345e8caec1513a02f
File name	135e31500f9ffb99d5d9e29ddd5bc36f_1.apk
File size	4.71 MB
Last analysis	2015-03-08 11:21:13 UTC

Detection | Details | Behavior | Community

Ad-Aware	Trojan.DOS.WMI	ALYac	Trojan.DOS.WMI
Avast	BV:ExitWindows-K [Trj]	AVG	Android_dc.AEDL
Avira	Android/Agent.A.904	AVware	Trojan.AndroidOS.Generic.A
BitDefender	Trojan.DOS.WMI	CAT-QuickHeal	BAT/CopyToAutoexec
ClamAV	W97M.Mary.A	Comodo	UnclassifiedMalware
Cyren	AndroidOS/GenBl.135E3150!Olympus	Emsisoft	Trojan.DOS.WMI (B)
eScan	Trojan.DOS.WMI	F-Secure	Trojan.DOS.WMI
GData	Trojan.DOS.WMI	Ikarus	Virus.BAT.Rbtg
Kingsoft	Android.HACKTOOL.at_HackBook.a. (kcloud)	McAfee	Artemis!135E31500F9F
McAfee-GW-Edition	Univ.script/99a	Microsoft	Virus:BAT/Rbtg.gen
NANO-Antivirus	Virus.Script.Ankit.gdgh	Symantec	Unix.Penguin
TheHacker	Bat/generic	TrendMicro	AndroidOS_Gen.D
TrendMicro-HouseCall	AndroidOS_Gen.D	VBA32	Virus.BAT.Winupdt.a
VIPRE	Trojan.AndroidOS.Generic.A	Zoner	Exploit.AndroidOS.Lotoor.A



Analyze your files with Metashield Clean-up Online.

To analyze the metadata in a file, choose the file below and click on "Analyze". Once you accept the Terms and Conditions of Use for Metashield Clean-up Online servicea screen will appear with the summary of metadata found.

dr4gchap4.rtf Select Analyze

Metadata found in **dr4gchap4.rtf**

- CharacterSet
- Comments
- InternalVersionNumber
- Languages
  - Values
    - Arabic (Saudi Arabia)
  - Categories
  - dr4gchap4.rtf

Hire *Metashield Clean-up online* to additionally clean the metadata of your documents in a final and simple way. Purchase

@unapibageek - @ssantosv

STRINGS

./aapt d --values strings android_app.apk

```
String #1029: Ver tudo
String #1030: Limpar consulta
String #1031: Navegar para cima
String #1032: Pesquisar...
String #1033: Pesquisar
String #1034: Válasszon ki egy alkalmazá
String #1035: Összecsuk
String #1036: Megosztás a következővel:
String #1037: Keresési lekérdez
String #1038: KI
String #1039: Összes megtekinté
String #1040: Felfelé mozgat
String #1041: További lehetősé
String #1042: Lekérdezés küld
String #1043: Kés
String #1044: Ugrás a főoldal
String #1045: Lekérdezés törl
String #1046: Hangalapú keres
String #1047: Keresés
String #1048: Megosztás a következőv
String #1049: Keresé
String #1050: BE
String #1051: Показа
String #1052: Свер
String #1053: Отправит
String #1054: Голосов
String #1055: Удалить
String #1056: Выбрать п
```

STRINGS

```
./aapt d --values resources android_app.apk
| grep '^ *resource.*:string/' --after-context=1 > output.txt
```

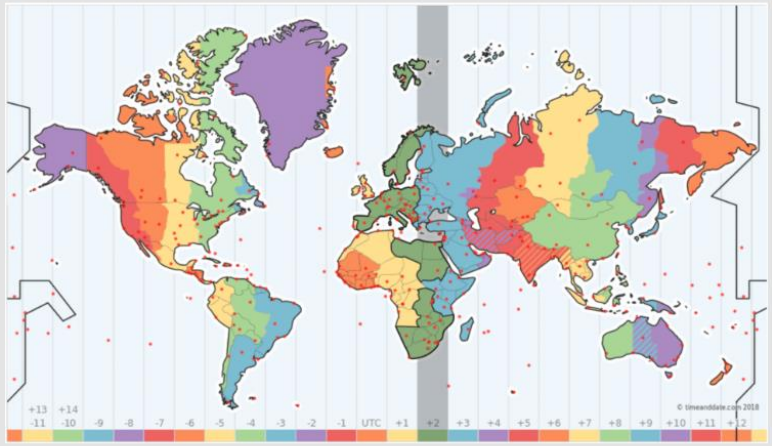
```
resource 0x7f06001d com.elevenpaths.shei.mytestgmt1:string/abc_font_family_headline_material  t=0x03
d=0x000001a0 (s=0x0008 r=0x00)
  (string8) "sans-serif"
resource 0x7f06001e com.elevenpaths.shei.mytestgmt1:string/abc_font_family_menu_material  t=0x03
d=0x000001a0 (s=0x0008 r=0x00)
  (string8) "sans-serif"
resource 0x7f06001f com.elevenpaths.shei.mytestgmt1:string/abc_font_family_subhead_material: t=0x03
d=0x000001a0 (s=0x0008 r=0x00)
  (string8) "sans-serif"
resource 0x7f060020 com.elevenpaths.shei.mytestgmt1:string/abc_font_family_title_material: t=0x03
d=0x000001a1 (s=0x0008 r=0x00)
  (string8) "sans-serif-medium"
resource 0x7f060021 com.elevenpaths.shei.mytestgmt1:string/app_name: t=0x03 d=0x000001a3 (s=0x0008 r=0x00)
  (string8) "MiTest"
resource 0x7f060022 com.elevenpaths.shei.mytestgmt1:string/app_name3: t=0x03 d=0x000001a4 (s=0x0008 r=0x00)
  (string8) "El idioma nativo del desarrollador es español"
resource 0x7f060023 com.elevenpaths.shei.mytestgmt1:string/app_name4: t=0x03 d=0x000001a5 (s=0x0008 r=0x00)
  (string8) "Ahora tu lo sabes"
resource 0x7f060024 com.elevenpaths.shei.mytestgmt1:string/app_name5: t=0x03 d=0x000001a6 (s=0x0008 r=0x00)
  (string8) "Una string mas"
resource 0x7f060025 com.elevenpaths.shei.mytestgmt1:string/name1: t=0x03 d=0x000001a7 (s=0x0008 r=0x00)
  (string8) "Esto es una cadena de texto escrita por el desarrollador"
```

OUR TOOL

Language info

STRINGS_XML	en:0.538 fr:0.064 da:0.044
XML_FILES_INFO	0:EN 1:IT 2:ES 3:ZH 4:RU 5:ZH 6:TR 7:DE

Summary



POSSIBLE GMT	2

Certificate

@unapibageek - @ssantosv

CONCLUSIONS & FUTURE WORK

We presented different ways for not just leaking timezone but as well...

- Possibly detecting automated malware creation
- Possible better Machine learning features in detecting APK malware
- A tool for a quick view of all this useful information around APKs metadata

Future work should be more accurate about DST and using more samples

THANK YOU!

Sheila Ayelen Berta

Security Researcher – ElevenPaths
(Telefonica Digital cyber security unit)
@UnaPibaGeek
sheila.berta@11paths.com

Sergio De Los Santos

Head of Research – ElevenPaths
(Telefonica Digital cyber security unit)
@ssantosv
sergio.delossantos@11paths.com