Toast Notifications

Alright ToastNotifications, why do we want to use them within Forensics or Incident Response? What is the forensic value of these notifications? Before we jump into answering these questions, let's take a brief look at what ToastNotifications actually are. Microsoft states the following:
*The notification area is a portion of the taskbar that provides a temporary source for notifications and status. It can also be used to display icons for system and program features that have no presence on the desktop, such as battery level, volume control, and network status. The notification area has been known historically as the system tray or status area. (*https://docs.microsoft.com/en-us/windows/win32/shell/notification-area)

Are notifications new? No, prior to Windows 10 notifications handled by the Windows operating system were displayed in the Action Center. With the introduction of Windows 10 this was changed to Notification Center. The notifications and alerts generated by the Windows operation system are so called toast notifications. A toast notifications is basically a pop-up on the users' screen, which will be active for several seconds on screen (depending on the preferences of the application). These toast notifications can be initialized by native Windows applications such as email, updates but also by third-party applications.
After the notifications are displayed on the desktop of the user, the notifications will be grouped inside the Notification Center up until they are 'cleaned up' by the user. A user can decide to turn off notifications of programs or entire notification system.

Forensic relevance and value

As explained above, both Windows native as well as third-party applications can generate toast notifications to notify the end user. This means, from a forensics perspective it could provide to be a source of evidence that potentially sketches activity of all system wide applications. This means that a Forensic examiner can determine how the system is used, what applications were running at both the moment of capture as well as in the past, and finally what interesting activity was so important the user had to be notified. Especially in case of the latter, this is key evidence for a forensic examiner during an investigation. Let us give you some examples:

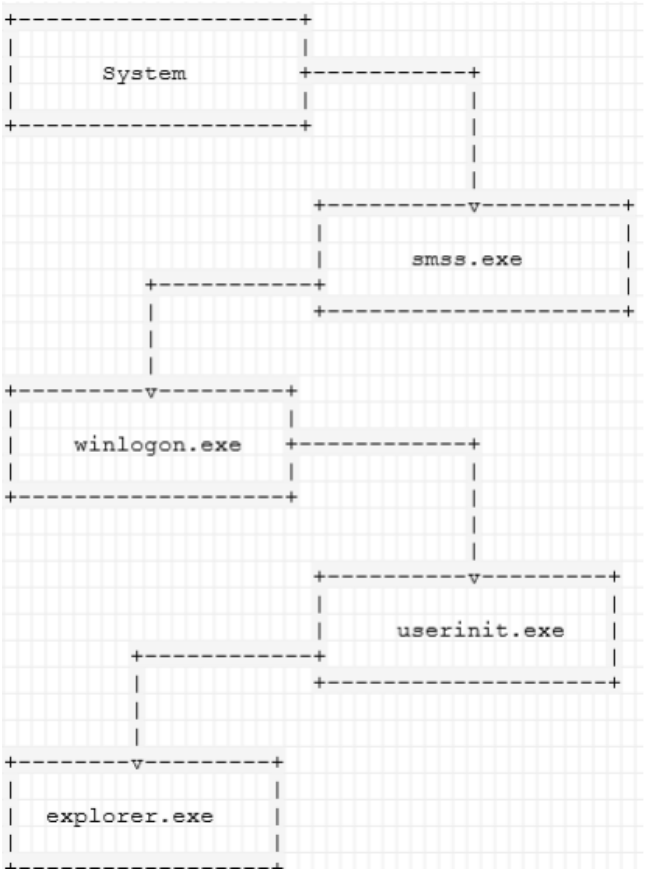
- Connected devices
- Receiving application or system critical alerts
- Receiving alerts of Antivirus solutions
- Receiving alerts of messages such as WhatsApp or even e-mail

However, when taking connecting devices as an example, one could argue we have the Windows registry to determine that? Alerts of the AV solution, it could be stated we have Windows event logs for that? True, but aren't we always looking for more artifacts to substantiate other artefacts? Also, these artefacts are well known by adversaries. Therefore, in the case of anti-forensics there is a good chance they might not think about ToastNotifications (up until now, after the release of this write-up ;)). We are convinced that these ToastNotifications can be of great forensic value since they allow an investigator to validate evidence.

Besides volatile artefacts of toast notifications there are also traces on disk in the wpndatabase.db sqlitedatabase, which is also retrievable from memory. The distinctions of our ToastPlugin is the fact that we scrape the ToastNotifications structure directly out of memory instead of finding a database file and parsing the content. We find it more likely that an adversary could have tampered with the SQLite database instead of the actual ToastNotifications structure found in memory. In addition, both artifacts could be a good combination and differences can be easily spotted.

Technical Details

To pop a toast notification a program needs to send the toast data via window messages to the window with the name `ShellTrayWHD`. This window is part of the explorer process that further handles the requests of toast notifications. The explorer process will check what type of data is embedded in the notification and chooses an xml template accordingly. This template is populated with the following function `UpdateToastTemplateWithData`. Following this, the toast notification is dispatched and shown to the user.



Looking at the structure of the ToastNotifications in memory, we found out that the XML structure is easily recognized. The image you will find below is an example of the XML structure observed in memory during the research.

```
<toast>
  <visual>
    <binding template='ToastText04'>
      <text id='1'>My app title</text>
      <text id='2'>My first text string</text>
      <text id='3'>My second text string</text>
    </binding>
  </visual>
  <actions>
    <action activationType='foreground' content='my action' arguments='an argument'/>
  </actions>
```

The XML structure contains all the information that's needed to visualize its contents. Information such as title and content are provided, also the location of files such as pictures or sound are embedded within this structure.
More info and details about the toast notification XML tags can be found on the Microsoft website: https://docs.microsoft.com/en-us/windows/uwp/design/shell/tiles-and-notifications/toast-schema
Unfortunately information such as the time of creation and duration of the notification are not present in this structure. In order to retrieve this, further research is needed on the explorer process.

ToastPlugin

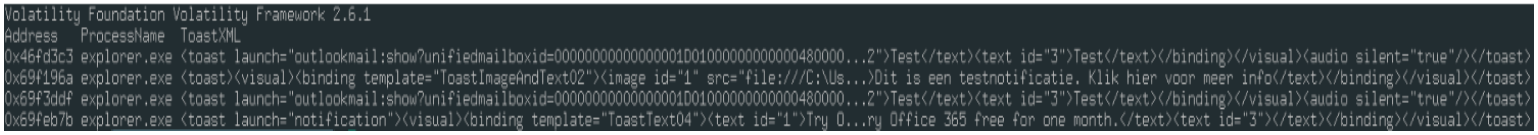
The toastplugin uses a regular expression and utilizes the yarascan plugin (https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal#yarascan) to find toast XML structures in memory. When YARA finds a hit the plugin parses the XML structure and displays it on the command line.
Running the volatility yaraplugin with the regex: /<toast.*\/toast>/ we find hits on the Start tags found in memory. We can utilize the yarascanplugin or the volshell plugin to jump to this address and analyze the content of the different tags.
Below you can find an illustration depicting a hit of a received email found by the yarascanplugin with our regex. We've highlighted some parts, where green highlights one of the unique tags to indicate the start of the ToastNotifications. In red and blue interesting forensic values of the ToastNotification are highlighted, in which case red highlights the actual content of the ToastNotificaton whereas blue highlights the indicators of the application and corresponding artifacts.

```
0x05e5d340  74 6f 61 73 74 20 6c 61 75 6e 63 68 3d 22 6f 75  toast.launch="ou
0x05e5d350  74 6c 6f 6f 6b 6d 61 69 6c 3a 73 68 6f 77 3f 75  tlookmail:show?u
0x05e5d360  6e 69 66 69 65 64 6d 61 69 6c 62 6f 78 69 64 3d  nifiedmailboxid=
0x05e5d370  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
0x05e5d380  31 44 30 31 30 30 30 30 30 30 30 30 30 30 30 30  1D0100000000000
0x05e5d390  34 38 30 30 30 30 30 30 31 39 30 30 30 30 30 30  4800000019000000
0x05e5d3a0  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  0000000000000000
0x05e5d3b0  31 44 30 31 30 30 30 30 30 30 30 30 30 30 30 30  1D0100000000000
0x05e5d3c0  46 45 46 46 46 46 46 46 46 46 46 46 46 46 46 46  FEFFFFFFFFFFFFFF
0x05e5d3d0  26 61 6d 70 3b 61 63 63 6f 75 6e 74 69 64 3d 30  &accountid=0
0x05e5d3e0  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 35  0000000000000005
0x05e5d3f0  39 30 31 30 30 30 30 30 30 30 30 30 30 30 30 34  9010000000000004

0x05e5d5a0  3e 3c 76 69 73 75 61 6c 20 6c 61 6e 67 3d 22 6e  ><visual.lang="n
0x05e5d5b0  6c 2d 4e 4c 22 3e 3c 62 69 6e 64 69 6e 67 20 74  l-NL"><binding.t
0x05e5d5c0  65 6d 70 6c 61 74 65 3d 22 54 6f 61 73 74 54 65  emplate="ToastTe
0x05e5d5d0  78 74 30 34 22 3e 3c 74 65 78 74 20 69 64 3d 22  xt04"><text.id="
0x05e5d5e0  31 22 3e 52 6f 6c 66 20 47 6f 76 65 72 73 3c 2f  1">Rolf.Govers</
0x05e5d5f0  74 65 78 74 3e 3c 74 65 78 74 20 69 64 3d 22 32  text><text.id="2
0x05e5d600  22 3e 74 65 73 74 3c 2f 74 65 78 74 3e 3c 74 65  ">test</text><te
0x05e5d610  78 74 20 69 64 3d 22 33 22 3e 74 65 73 74 3c 2f  xt.id="3">test</
0x05e5d620  74 65 78 74 3e 3c 2f 62 69 6e 64 69 6e 67 3e 3c  text></binding><
0x05e5d630  2f 76 69 73 75 61 6c 3e 3c 61 75 64 69 6f 20 73  /visual><audio.s
```

Figure 3: ToastNotfiicatoinXML structure in memory

What does this look like running the actual plugin? Below you will find example output of the plugin, which successfully found four ToastNotifiactions in memory. Of the aforementioned ToastNotifications three are initialized by outlook and 1 by a custom powershell script used for testing.



Future work

We call this the first stage of the plugin, we have done a lot of research already but not extensive enough to translate it into the plugin. We have drawn a road map and divided our development of the plugin into several stages:

1. Base plugin for Volatility Contest submission
2. Optimize and enrich parsing of output, to display on the CLI
3. Adding more metadata, like ArrivalTimestamp, ExpireTimestamp and application information
4. Combine evidence with the content of the wpndatabase.db found in memory to create a cross view.

Authors = Max de Bruijn , Rolf Govers
Department = Forensics and Incident Response
Company = Fox-IT B.V.
Year = 2019