Diving into Intel Killer bloatware, part 1

zwclose.github.io/2022/12/18/killer1.html

ZwClose December 18, 2022

Dec 18, 2022

Killer Control Center before version 2.4.3337.0 is prone to tampering (person-in-the-middle) attack. Remote attacker can start, stop, enable or disable any service and block network access for any process in the OS regardless of their privileges. Killer Control Center downloads unsigned configuration file from Killer's web server via plain HTTP. The configuration file contains, amongst other things, definitions of processes' network bandwidth limits and directives for Killer's Boost modes. Network limits allow to block a specific process from accessing network by process's image name. Boost modes define which services should be on or off when the certain mode is enabled. Lack of signature of the configuration file makes it possible for the person-in-the-middle to maniuplate bandwidth limits and services on the remote computer by adding/modyfing entries to the configuration file during the update process. The update should be triggered by user via Killer Control Center UI. Bandwidth limit definitions get applied immediately upon the update while service definitions, dependently on settings, might need enabling Gamefast mode from the UI.

Some OEMs, e.g. Dell, preinstall Killer Control Center to their laptops which significantly increases the impact of the vulnerability. The vulnerability existed undetected for a few years, I guess starting from 2016. Intel has confirmed and fixed it, CVE-2021-26258 was assigned.

In this blog post I cover the details of the vulnerability, provide a PoC and show the video of the attack.

Some time ago I bought a slick and shiny Dell XPS. Many people don't do fresh install after buying a laptop, so didn't I because I wanted to research how vulnerable the default software package is. The research revealed a few security issues in Killer suite.

Originally Killer was developed by Rivet Networks for Killer-branded network cards. The cards and the accompanying application intended to improve gaming experience, e.g. to lower ping, which might be critical for gaming. Some time later Rivet and Intel co-operatively released a few gaming oriented NICs. Finally, in 2020 Intel acquired Rivet Networks, so these days Killer Control Center belongs to Intel.

Killer performance suite provides a few features to shape network traffic. The most fruitful for us features are Prioritization engine and Gamefast mode. Prioritization engine allows to set network bandwidth limits for a specific process, say, svchost.exe while Gamefast

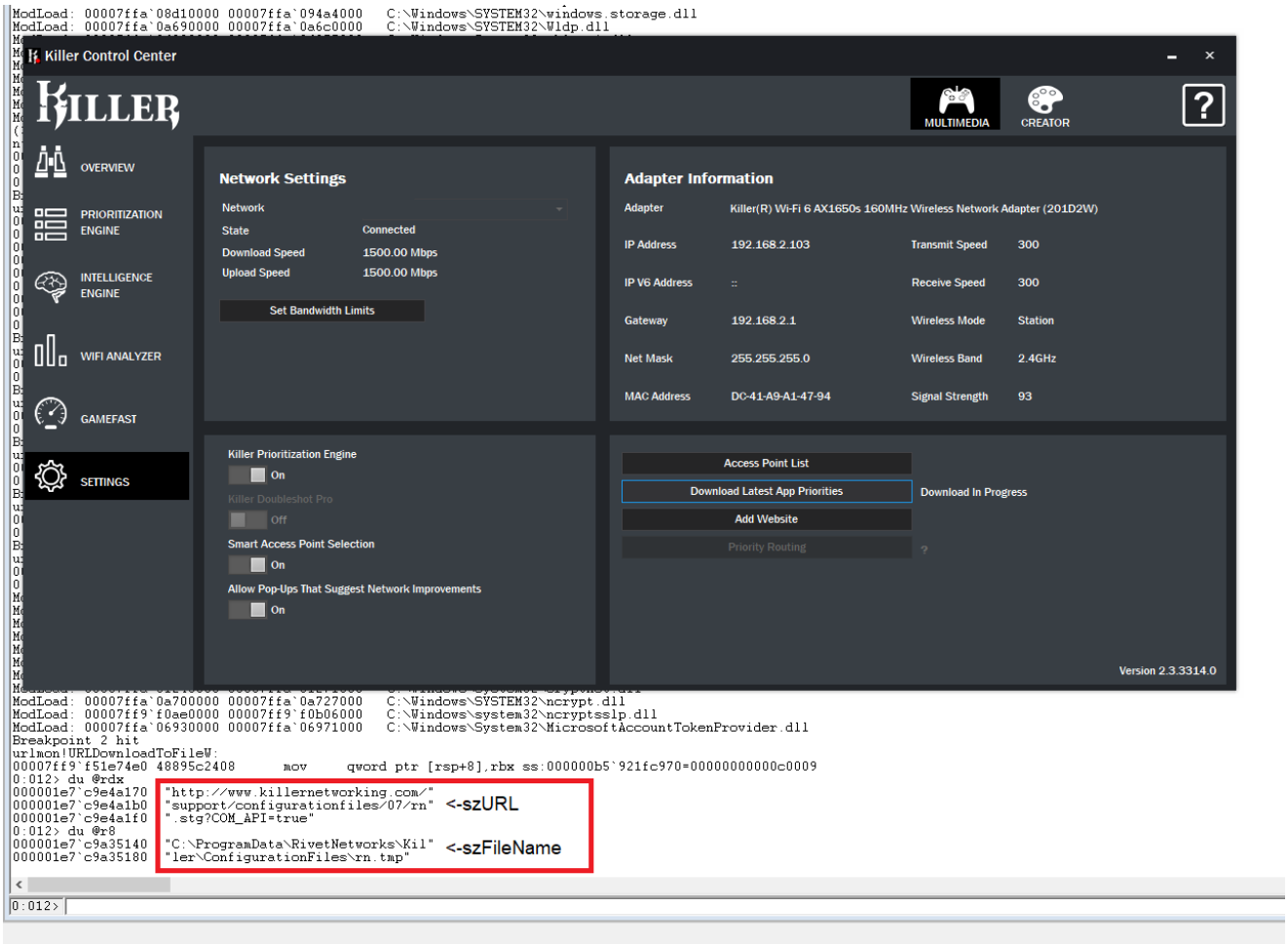
manipulates Windows services, e.g. enables and starts RemoteRegistry, once it detects execution of a gaming process. Definitions of what process is a gaming process, what to do with the serivces when it's running and processes' network bandwidth limits are stored rn.stg file located %ProgramData%\RivetNetworks\Killer\ConfigurationFiles\ folder. Killer's management service runs under NT AUTHORITY\System account and uses WFP driver to implement bandwidth policies, which gives it enough pribileges to disrupt critical services and security related software.

It looks that in 2016 Rivet decided to gather definitions for the best gaming experience in a huge common rn.stg file and propagate it via their web server. Users could download and apply the file by clicking "Download Latest App Priorities" button in Killer Control Center's UI. The problem is that the downloading wasn't secure enough: as said earlier the file wasn't signed and got downloaded via plain HTTP, hence it could be faked by a person in the middle. Lack of verification let the attacker to apply aforementioned traffic shaping to the remote system. As of May 2021 and later rn.stg was not available at Killer's web site but "Download Latest App Priorities" button was still present in the UI making the attack possible.

In the following sections I will go over unpacking of rn.stg and show how to modify it to start a random (even disabled) service and to block a process from accessing network. Full code used in the blogpost is available in git repository.

[Unpacking rn.stg]

When user clicks "Download Latest App Priorities" button Killer's backing service KillerNetworkService.exe calls URLDownloadToFile API to download rn.stg from the server to rn.tmp file.



Upon successful downloading KillerNetworkService verifies MD5 of rn.tmp, copies rn.tmp to rn.stg, which is a persistent storage of Killer's settings, and reloads rn.stg to adopt new definitions. rn.stg is a structed storage file that bears named stream rn.xml and user defined property MD5Checksum. The property contains the checksum of rn.xml stream and rn.xml stream in turn contains definitions for the prioritization engine. As it can be guessed from the stream name, the content of the stream is XML file. To make things less obvious, the content of rn.xml is encrypted with the most famous encryption operation ever: xor. Here's how the decryption code looks like in HexRays:

```
440             if ( pStreamBuf && key )          // de xor content
441             {
442               pXmlBuf = pStreamBuf;
443               length = v102 - pStreamBuf;
444               do
445               {
446                 LODWORD(key) = 0x10003 * key + 1;
447                 *pXmlBuf ^= BYTE2(key);
448                 ++pXmlBuf;
449                 --length;
450               }
451               while ( length );
452             }
```

The decrypted rn.xml is huge, here is an excerpt from it:

```
<?xml version="1.0" encoding="UTF-16"?>
<BWC>
  <!--10/27/15 Version 0.1 - Initial Check-in Development.-->
  <!--08/30/16 Version 1.0 - First Release To Public Config Files-->
  <!--01/12/17 Version 2.0 - Compressible XML Files-->
  <!--07/07/17 Version 3.0 - Categories are now in full force (no attributes when we
have a category)-->
  <!--08/25/17 Version 3.0 - No Change, but introduced new MultiInterfaceRoutingType
Classes - Redundant with MultiInterface will rev version when removing redundant
class-->
  <!--02/01/18 Version 4.0 - Adding ActivityRules and renaming Rules to
ProcessDomainRules-->
  <!--09/01/18 Version 5.0 - Adding BoostedGroups And now VersionCheck now ignores
Minor Versions differences for XML Files -->
  <!--12/07/18 Version 5.1 - Removed Redundant with MultiInterface Node- No Need to
rev version as Version 4 on software will not look for this old tag.-->
  <!--07/24/19 Version 5.2 - Added ContentMajor and ContentMinor to Version tag.-->
  <!--07/29/19 Version 5.2 - Changed to ContentVersion to Version tag.-->
  <!--09/06/19 Version 5.3 - Combined Killer and Creators and using Profiles.-->
  <!--10/16/19 Version 5.4 - Changed names of Profile Killer to Gaming, Categories
Creator/Games to Profile - Should up the major version as old app not compatible with
config file but service is.-->
  <!--10/17/19 Version 6.0 - Reverted Category Profile Change now back to
Creator/Games.  Also increasing Major Version as Service API/APP got upgraded
although not required by service but by App-->
  <!--02/28/20 Version 7.0 - Removed Current Profile Tag - code has default and also
now overridable in registry-->
  <!-- Also update Version in
KillerNetwork_Service\Service\SharedSource\inc\BWCXMLCommon.h-->
  <Version Major="7" Minor="0" ContentVersion="1.0.0"/>
  <Gaming>
    <Configuration Comment="Updated: 2020.09.08"/>
      <ActivityRules>
        <Rule Name="EdgeConditionMatch" Type="PortConnectionDisconnection">
          <Action Name ="ComputePriorityAndConnectionCount" Argc="0"/>
        </Rule>
      </ActivityRules>
      <BoostedGroups>
        <BoostedGroup Type="Category" Name="Games">
          <Success_Action>
            <Action Name="ServiceAction" Argc="4"
Argument1="AppleMobileDeviceService" Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="AJRouter" Argument2="4"
Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="DiagTrack" Argument2="4"
Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="Dmwappushservice"
Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="DPS" Argument2="4"
Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="DusmSvc" Argument2="4"
Argument3="1" Argument4="0"/>
```

```
            <Action Name="ServiceAction" Argc="4" Argument1="FrontCache"
Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="HomeGroupProvider"
Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4"
Argument1="KillerSmartConnectService" Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="MapsBroker"
Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="NcdAutoSetup"
Argument2="4" Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="PcaSvc" Argument2="4"
Argument3="1" Argument4="0"/>
            <Action Name="ServiceAction" Argc="4" Argument1="PlexUpdateService"
Argument2="4" Argument3="1" Argument4="0"/>
...
```

[Starting a service]

Gamefast feature tries to optimize memory usage by disabling or enabling certain services when user plays a game. If the feature is on, Killer monitors running processes to figure out if user plays a game. Once gaming process is detected, Killer goes over the list of services stored in rn.xml and applies the corresponding service action to every service in the list.

The database of services and the corresponding actions is stored under BoostedGroups node of rn.xml:

```
<BoostedGroups>
  <BoostedGroup Type="Category" Name="Games">
    <Success_Action>
      <Action Name="ServiceAction" Argc="4" Argument1="AppleMobileDeviceService"
Argument2="4" Argument3="1" Argument4="0"/>
      ...
```

Each Action node and its attributes represent a service and what shoud be done on it once a game is detected.

Attribute Argument1 designates name of the service in question. Argument2 is service's start type which is usually either *SERVICE_AUTO_START* (0x00000002) or *SERVICE_DISABLED* (0x00000004). Killer calls ChangeServiceConfigW API for the service passing to it the value of the attribute. It lets the attacker to enable or to disable any service in the OS:

```
80      if ( QueryServiceConfigW(hSvc, 0i64, 0, &pcbBytesNeeded)
81        || GetLastError() != 0x7A
82        || (sub_140021AF0(lpServiceConfig, pcbBytesNeeded),
83            v12 = lpServiceConfig[0],
84            v13 = QueryServiceConfigW(hSvc, lpServiceConfig[0], pcbBytesNeeded, &pcbBytesNeeded),
85            v6 = v13,
86            !v13)
87        || Argument2 != v12->dwStartType )
88      {
89        v6 = ChangeServiceConfigW(
90               hSvc,
91               0xFFFFFFFF,                      // dwServiceType, SERVICE_NO_CHANGE
92               Argument2,                       // dwStartType
93               0xFFFFFFFF,                      // dwErrorControl, SERVICE_NO_CHANGE
94               0i64,
95               0i64,
96               0i64,
97               0i64,
98               0i64,
99               0i64,
100              0i64);
101     }
```

Argument3 controls state of the service, which lets to start or to stop it remotely. Values 1, 3, 6, 7 stop the service whereas 2, 4, 5 start it:

```
26    switch ( Argument3 )
27    {
28      case 1u:
29      case 3u:
30      case 6u:
31      case 7u:
32        Argument3 = 1;
33        v9 = sub_14001E100(&v17, strSvcName);
34        v10 = OpenSCManagerW_Wrap(&v16);
35        v8 = StopSvc(v10, v9, 0x20u);
36        goto LABEL_6;
37      case 2u:
38      case 4u:
39      case 5u:
40        Argument3 = 4;
41        v11 = sub_14001E100(&v17, strSvcName);
42        v12 = OpenSCManagerW_Wrap(&v16);
43        v8 = StartSvc(v12, v11, 0x10u);
44LABEL_6:
45        CloseSvc(&v16);
46        break;
47      default:
48        break;
49    }
```

Finally, Argument4 tells KillerNetworkService wether it should wait until the service actually changes its state or return immediately. Setting this value to zero worked perfectly for me.

The following XML snippet enables and runs RemoteRegistry service:

```
<BoostedGroups>
  <BoostedGroup Type="Category" Name="Games">
    <Success_Action>
      <Action Name="ServiceAction" Argc="4" Argument1="RemoteRegistry" Argument2="2"
Argument3="2" Argument4="0"/>
    </Success_Action>
  </BoostedGroup>
</BoostedGroups>
```

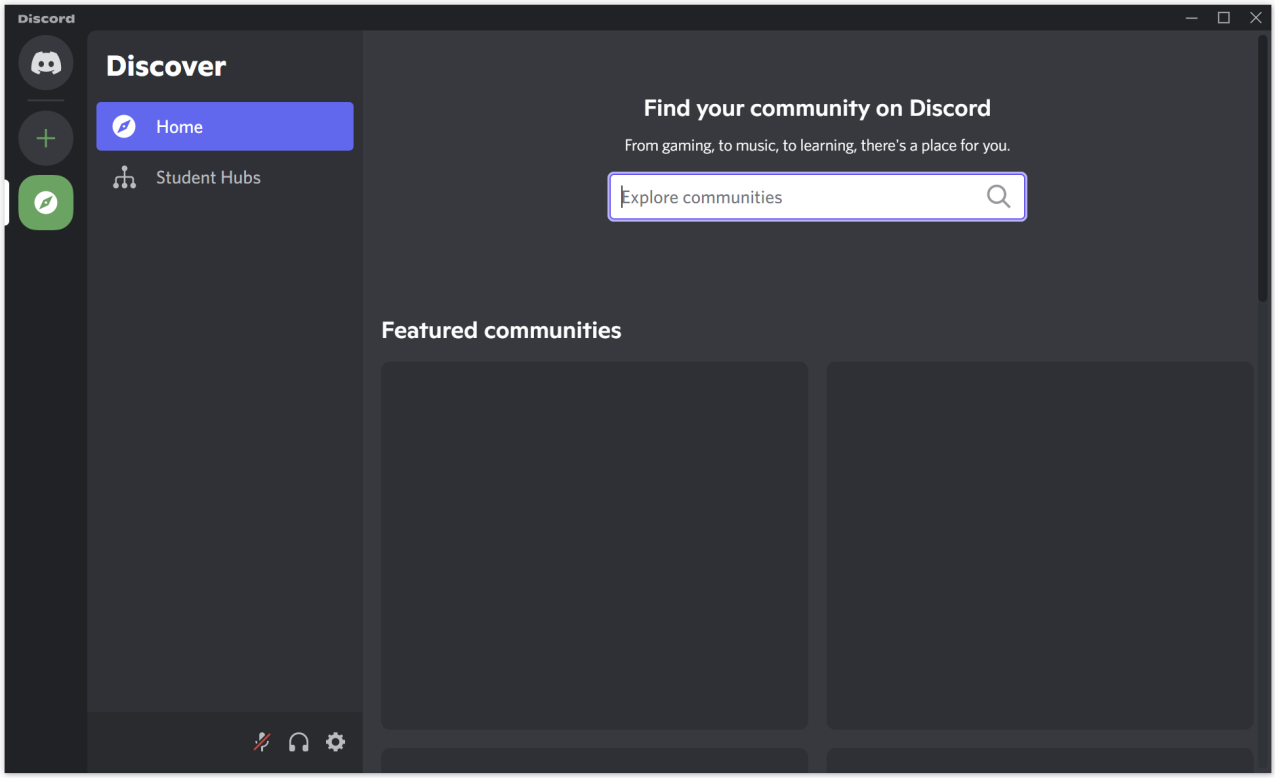
[Blocking a process]

Prioritization engine allows to shape network traffic by setting network bandwidth limits for a specific process. To enforce bandwidth policies Killer uses Windows Filtering Platform driver named KfeCo10X64.sys, which allows to block network access for a process regardless of its privileges. The list of processes and their network limits is stored in rn.xml under ProcessDomainRules node. The list consists of Rule child nodes each of them representing a process to be limited. It is easier to explain the structure of Rule node with an example. The following XML snippet disables both upload and download traffic for Discrod.exe by setting BandwidthUp and BandwidthDown attributes to zero.

```
<ProcessDomainRules>
  <Rule Name="Match By Name" Type="Process">
    <Action Name="MatchName" Argc="1" Argument1=".*\\Discord\.exe" />
    <Success_Action>
      <Action Name="SetAttribute" Argc="2" Argument1="Category" Argument2="Downloads"
/>
      <Action Name="SetAttribute" Argc="2" Argument1="BandwidthUp" Argument2="0" />
      <Action Name="SetAttribute" Argc="2" Argument1="BandwidthDown" Argument2="0" />
    </Success_Action>
  </Rule>
</ProcessDomainRules>
```

It worth to note that Killer identifies processes using their pathnames and uses wild characters to make identification more flexible. Setting Argument1 attribute in the snippet above to ".*\Discord.exe" applies the limits to all Discord.exe processes regardless of their root directory. If you rename a random executable to Discord.exe and run it, the process won't be able to perform network communication just because of its image name.

Unlike Gamefast, prioritization engine is enabled by default and doesn't require any user interaction. The database of network limits gets applied immediately once rn.stg is updated. The pic below shows how networkless Discord looks like.



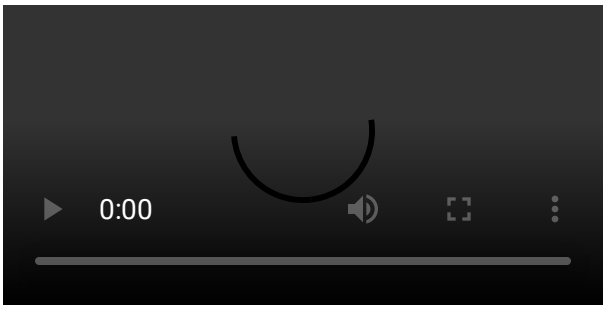
[Demo]

Now we have everything to assemble XML snippets scattered in the post into single rn.stg file and run our demo.

First of all we need to create custom rn.stg with a stream named rn.xml and a property named MD5Checksum. Then we need to encrypt our custom XML file, calculate its MD5, write the encrypted content to rn.xml stream and write MD5 to MD5Checksum property. The source code of the application that extracts XML content from rn.stg and creates rn.stg from XML file is available in the repository.

Second, we need to set up the environment that imitates person in the middle attack. In the real life such attack can be done, say, via rogue WiFi spot, but for simplicity let's use a tiny web server written in Python. The web server doesn't do anything but serves all HTTP requests with custom rn.stg that should be placed in the same directory as the server. The source code of the server is also available at github. To redirect requests from killernetworking.com to the local web server add the following entry to hosts file: 127.0.0.1 www.killernetworking.com

Once the web server and hosts file are set we are ready to click "Download Latest App Priorities" button in Killer's UI to download custom rn.stg file. Normally downloading would fail because the original rn.stg is moved from Killer's server but in case of attack downloading should succeed, which means that Killer fetched and parsed rogue rn.stg. Here is the video of the attack:



[Disclosure]

Disclosure process wasn't that straightforward. I spotted the vulnerability in June 2021. It took me some time to figure out details and submit them to Intel. We exchanged a few messages and in October 2021 they replied that "the vulnerability was present in the software when Killer was still owned by Rivet and the first version of the software under the Intel brand did not have this vulnerability. Intel never posted or distributed a version of the software that has this vulnerability." Despite of that Intel awarded me with 1500 USD of bug bounty. Yet I felt a bit perplexed: what is it, an orphaned vuln? Or is it Dell's issue? I informed Dell about the vulnerability and soon after that they released an update that fixed Killer Control Center. I then messaged Dell again to make sure I can disclose. They replied that Intel has confirmed the vuln and now want to make sure that other OEMs have adopted the update, which was a bit weird for a vuln that was not present anywhere. Anyway, the disclosure was postponed to February 2022 and then to May 2022. Finally, on May 10 Intel released the SA and Dell released the DSA. Long way, ah?

Disclosure timeline:

07/2021: initial discovery

01/09/2021 me->Intel: fix your stuff

07/10/2021 me<-Intel: not ours
08/10/2021 me: ???
08/10/2021 me->Dell: Dell, yours?

02/11/2021 me<-Dell: Intell will fix it by Feb 8

10/02/2021 me->Dell: ready?
11/02/2021 me<-Dell: Intel will fix it by May 10

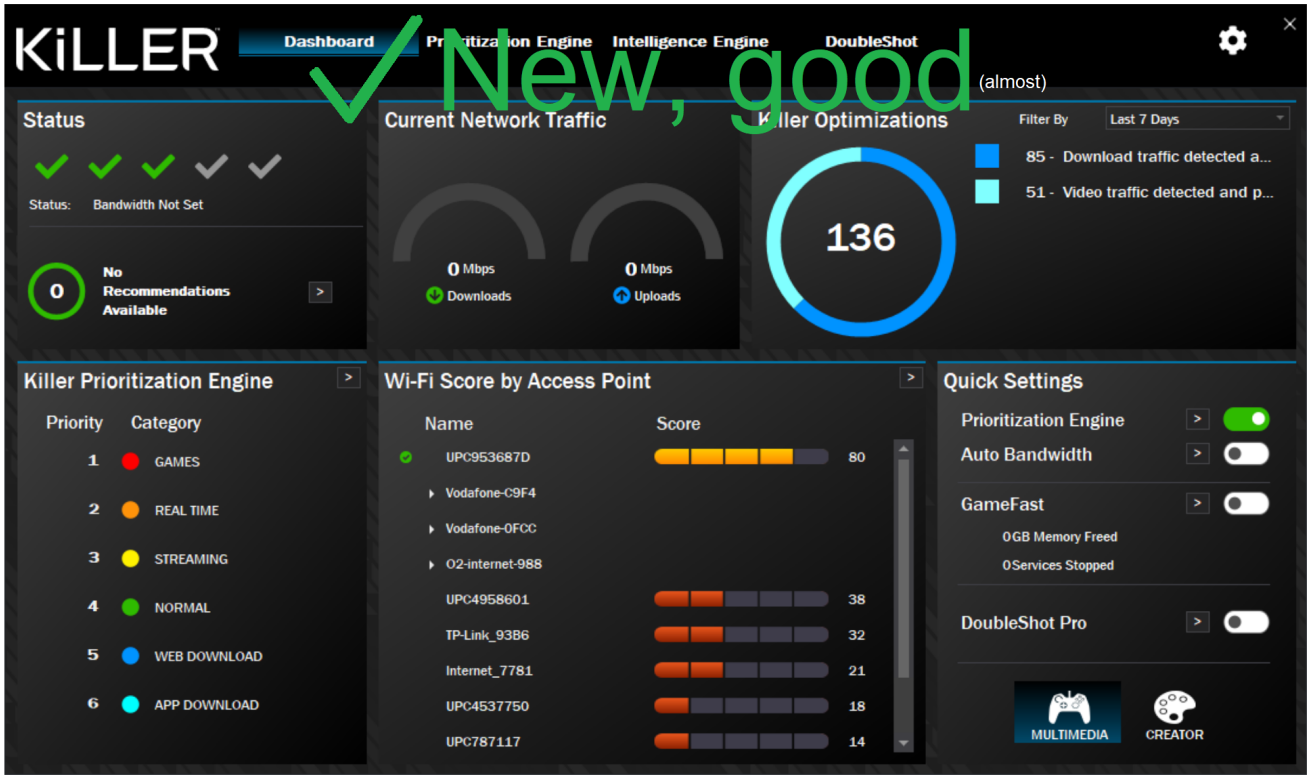
18/02/2021 me<-Intel: we will fix it by May 10

10/05/2021 vuln fixed, advisory published

[To be concluded]

So, it took more than six months for Intel (or Dell?) to fix Killer Control Center. Also, it got renamed to Killer Intelligence Center. I didn't figure out if it was Dell issue or Intel issue.

Dell is not the only OEM that preinstalls Killer, so does MSI, Acer, maybe other vendors too. I didn't check if they use fixed version of Killer but I hope that Intel have notified relevant OEMs. There is simple way to know if you are using safe version of Killer from its look: GUI of the fixed version is very different from the old one:



The story doesn't end here. Some time later I found another vulnerablity in Killer suite that affects way more OEMs and reported it to Intel. Intel never replied to me but released a quiet patch that fixed it. I also reported the vuln to Mitre to get a CVE ID but, guess what, Intel have their own CNA, so the CVE request got ignored too. Since the vuln is patched I feel free to disclose it in the next post.