security

September 15, 2020

1 Security

This document is completing the public, user guide Security. It is defining the implementation of the Security Policy, in more technical terms not necessary relevant to the end user but important for the developer to understand and to agree upon.

1.1 Overlay functions

DashT is a plug-in of OpenCPN and is binded to it through an API and an on-init shared library linking mechanism. From the security point of view, it has two functional blocks:

1. The classical,compiled plug-in part which is a part of the OpenCPN application what comes to the security analysis:

- It is using a binary ABI (application binary interface `opencpn.lib`) defined by an API ;
- It can create widgets onto the canvas using wxWidgets library functions ;
- It can create an graphical overlay onto the canvas using OpenGL ;
- It can read and write routing and waypoint information via the ABI ;
- It can read, create and modify files in the local file system via wxWidgets library while remaining identified as "OpenCPN".

2. Floating browser technology instruments loosely coupled to "OpenCPN" process

- OpenCPN provides the window frame;
- the execution of the JavaScript code via the wxWidgets library belonging equally to "OpenCPN" process ;
- JavaScript codes communicates with *DashT* via another, JavaScript text based API proper to *DashT*.

1.2 Data flow

See the corresponding user's guide section.

1.3 Security Policy

See the corresponding user's guide section.

1.4 Security Implementation

The interfaces are categorized first by the threat level they present to inject malicious code and to modify the navigation or other data. An implementation plan is defined for all interface threats.

1. **Signal K server node delta channel - LOW**

- *Risk Assessment*: This is a TCP/IP socket with text based protocol (Signal K / JSON). Since it is based on a subscription, its usage is low volume and data is parsed character by character in various level of parsers, first in JSON parser and then in Signal K parser. Both parsers are incorporated in *DashT* code base and constantly passed through an external security vulnerability parser ;
- *Implementation*: Several vulnerabilities, such as non-zero ending strings and similar have been fixed after the reports of those from the external advisory ;
- *Evolution* : In case of the inactivity of the development, no commits will be made to trigger the inspection and the external advisories will not be able to inspect the existing code against the new, detected vulnerabilities.

2. **OpenCPN Plug-In ABI - HIGH**

- *Risk Assessment*: The API implementation (ABI) is out of configuration control of *DashT*. It has been developed, in organic manner during the entire lifespan of *OpenCPN* to meet the functional requirements. There is no security policy or security implementation plan available when this is made. However, the code base is entirely available and inspectable: the external advisory reports several issues in the API, thus applicable but which cannot be unfortunately fixed in the provided ABI. Most serious - quite surprisingly - is the CWE-126 Buffer Over-read but there are others ;
- *Implementation*: The *DashT* code base addressing and using the *OpenCPN* ABI is regularly scanned against the very same vulnerabilities and they are all fixed and the scan is constantly maintained in every commit. When applicable, a copy of the data retrieved from *OpenCPN* is made and passed through the *DashT* code on which the security fixes have been made ;
- *Evolution* : It is likely but not guaranteed that *OpenCPN* community publish a security policy and starts implementing fixes to the code which creates a security vulnerability report. Meanwhile, the same inactivity risk of the *DashT* code itself will remain - more there is development, more there will be security fixes. There is no guarantee that this is a constant activity in this regard.

3. **InfluxDB v2 - LOW**

- *Risk Assessment*: InfluxData is a company which has huge number of clients, both paying and free around the world. They have a clear, world-wide security reporting policy ;
- *Implementation*: *DashT* does not incorporate any of the third party software but asks the end user to download and installs it. This makes sure that user gets also the latest security updates ;
- *Evolution* The risk can be in the evolution of the client side, interfacing software. This is copied as a snapshot from repositories of InfluxData. It may occur that a security issue is detected in those and we would not be necessarily aware of the fix. However, since this code is integrated, it passes also the consant integration security checks and it is likely that the vulnerability be eventually detected. For this, the development must be continuous.

4. **C++ / JavaScript instrument API - VERY LOW**

- *Risk Assessment*: This is a proprietary, text base API. In both sides text parsing takes into account the applicable security advisory for both programming platforms ;
- *Implementation*: The parsing code and command implementing code is passed through the external security advisory during continuous integration - the TypeScript / JavaScript code is passed on two, independent platform. In addition, the *node_modules* development environment is also checked against vulnerable development modules at each commit. It must be noted that none of this code is addressing external, unknown servers ;

- *Evolution* : The threats in web technologies are moving ahead quickly. A pause in the development usually causes an avalanche of reports next time the development will start again. This can go over the capacity of the developer to deal with all proposed fixes and they may be ignored.

As described in the Section 1.2, the code base is divided in two, both functionally and by programming environment:

1. C++ code running in *OpenCPN* process space - with independent threads running for input / output communications ;
2. TypeScript / JavaScript code running in *OpenCPN* process space, interfaced with *libwx_gtk3u_webview_3.0* library.

This allows better implementation when addressing the assessed risks. For example, if there is a need to address third party applications over networks, they do not have to be done in C++ code base which can be potentially be compromised via the ABI - or which can potentially be risk to the *OpenCPN* via the very same ABI.

The usage of TypeScript over JavaScript is not reducing the risk of the injection type of vulnerabilities *per se* - the transpiled code is still JavaScript. But stronger typing and the better code quality which follows is a security asset. Typed parameters, used together with a Mozilla project originated *Sanitizer* to strip potential attack vectors reduces further the risk of hacker being able to penetrate into the system by injection.

The segregation between two programming environments, one compiled and the other transpiled and then interpreted are connected via a proprietary, text based API. This greatly increases the segregation.

> *Risk Assessment*: users who build themselves *DashT* from this code base for Linux based system may enable their systems with *libwx_gtk***2***u_webview_3.0* : this GTK2 library is not supported anymore and is thus a security risk. For this reason, *DashT* official version does not support GTK2 based systems, only GTK3.

Each commit which is used in the published, official version is signed by the certified author(s).