DRAFT

Standard for Application Behavior Manifest

(E.O. Section 4(I)- Guidelines outlining security measures that shall be applied to the federal government's use of critical software)

S.Davis (Splunk)

Abstract

Recent software supply chain compromises have shown that we need a more robust framework to help restrict applications to known good behaviors, and then detect when applications deviate from said behaviors. To date, this has been done through application installation/configuration guides that help to define what the application should be doing. This is typically in the form of domains/IP addresses/protocols/ports that the application needs to communicate with. Quite often, this is a best effort from the software vendor, and can quite often change after publication.

The purpose of this submission is to propose a standards-based application behavior manifest that is provided by vendors for their software. The manifest would be published by the vendor, and would be a version controlled artifact. This would allow consumers of the application to use the version of the manifest that applies to the version of software they are using.

Introduction

The current approach to defining how applications behave is severely lacking. This usually involves an installation or configuration guide that provides port and protocol information. And depending on the level of complexity of the application, this list can stretch out to many pages. Quite often, network level controls are very relaxed for the sake of getting the application to work, or due to the documentation not being sufficiently clear.

When the application is then installed on a system that requires other services to function, this further increases the complexity. If process X needs access to service Y, process Z would typically be allowed as well.

Having a standard application manifest (I've been calling this an application "Hallpass"), would provide both network and application level definitions for what is considered good behavior for the application. The application "Hallpass" would be used by protection technologies like firewalls, proxies, and application control technologies to easily define their rulesets. It would also be used by detection technologies like SIEMs to alert on any behavior that deviates from the definition.

Design

Using a format such as JSON or XML would provide a standard hierarchical structure to build upon. A JSON example could look like (truncated for brevity):

```
{
   Vendor/product definitions here (software, version, etc),
    "type": "network",
       "inbound": [
          {
             Network level definitions here (addresses, ports, etc)
             }
          },
       ],
       "outbound": [
          {
             Network level definitions here (addresses, ports, etc)
             }
          }
        ],
    "type": "endpoint",
       "inbound": [
          {
             Application/process level definitions here (processes, addresses, ports, etc)
             }
          },
       ],
       "outbound": [
          {
             Application/process level definitions here (processes, addresses, ports, etc)
             }
          }
    ]
}
```

Conclusion

A standards-based approach to defining how applications communicate will help the industry to better protect and detect when things aren't behaving as expected. By having a living definition that can be published by the software vendor (APIs could be used to dynamically provide the "Hallpass"), controls and detections can adapt as applications change. And by making application security configurations simple to deploy, human error (and laziness) can be avoided.