AI Solution for Cyber Security

Cyber Security is about keeping the information in an enterprise safe. Leaks can happen as a result of malicious attack, or sometimes just careless, noncompliant actions from staff during routine work. System security software aims to detect any such activities which can represent an anomaly in a business workflow, violation of security protocol, or outright malicious external attacks.

Watching all applications running on all kinds of operating systems, with a diverse range of third-party applications installed and executing is anything but a straightforward proposition. A security software agent injects monitoring and control by keeping tabs on the operating system internals, meaning basic operations such as file read / writes, network connections, process start / stops, etc.

The Blue Screen of Death Dilemma

Although there is the idea of being non-invasive, the agent's use of process injection may create serious faults that could crash the application being monitored. It becomes a perception problem for a cyber security product in that every time a user sees a blue screen, it's easy to just blame the monitoring software. For a security software producer, these user complaints or suspicions cannot be taken lightly, especially at least some of these suspicions will prove to be valid. Even when a crash happens due to a completely different reason, the security software company still must do sufficient due diligence to eliminate it's software agent as the culprit.

Ideally, the producer would like to, by release time, know every single scenario where the agent interacts with some third-party application in an unfriendly way, resulting in a crash. However, given the number of potential applications and versions that could co-exist in the same operating system, It's quite impossible to rely on a comprehensive test plan to cover the entire spectrum of os-agent-3rd party app combinations. Most likely, the company will discover an errand agent behavior from repeating customer complaints, with heavy penalties levied on the product's reputation.

BigR.io Services

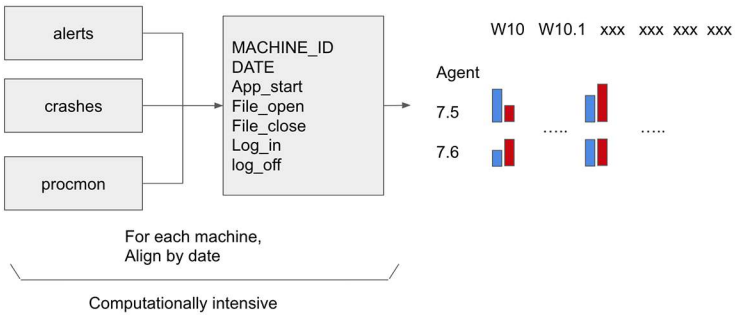
Here is where our statistical analysis can make a significant difference. For example, by collecting all the crash data and matching them against installed apps, we make it possible to detect the level of correlation between each pair of apps, revealing any possible cause and effect relation between them. By experimenting specifically with a suspected pair combination, we would be making it possible to remove the agent software as the origin of the conflict. It is also of interest to inspect whether any high correlation instance persists across os-agent version configurations, and even across tenants. Each result tells whether the conflict exists widely or remains isolated.

Correlation Study

Our engagement with a cyber security software company aimed at exactly this exercise. The company's agent collects various alerts, crashes, and process information over time. These are aligned with other collected data on application crashes, ci-resident installed apps and machines that host them. Our analytic code sorts all the raw data into a data frame consisting of rows of installed-crashed app pairs. Both the size and complexity of the calculation necessitates acceleration via multi-core parallel processing. The final report has a ranked list of app pairs with their correlation scores, which can, in turn, breakdown into os-agent version cells. With the insights from the correlation charts, the engineering team can take the highest ranking app pairs and validate the relative likelihood of crashes both with and without the presence of the client company's software agent.

Aggregation techniques reveals the correlation between installed and crashed apps for various machine environments.

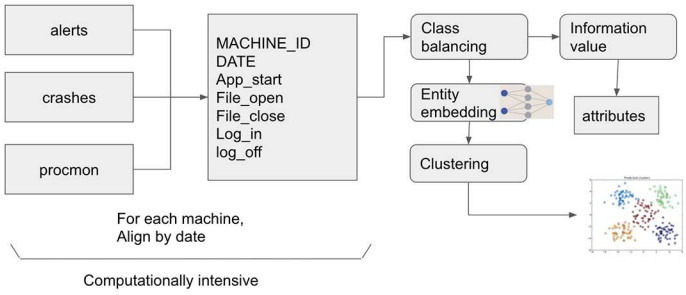
Install-Crash Correlation Data Flow



A Clustering Analysis

Moving into the next level of analysis, it is possible to apply clustering to the data and visualize whether there are obvious group behaviors among certain alerts, while incorporating a comprehensive list of attributes, including os, agent-version, as well as the installed / crashed app data. Given the extraordinary number of possible apps, the process would first go through encoding and dimensional reduction in a preparatory stage. The os and agent-version vectors consist of discrete string values that do not directly yield distance information. However, they can be reduced to numerical vectors by what's called entity embedding. The resulting numerical vectors would actually retain meaning semantics, whereby os versions that behave similarly with regard to alerts would be in close proximity when plotted on a cluster diagram. It would be interesting, for an example, if for some reason, Windows 10 early releases all huddle closely, but displays a clear separation from later releases at a particular build number. It provides insights to guide further analysis.

Cluster Data Flow



With all the attributes reduced to numerical vectors of reasonable dimensions, the reduced data set can then be clustered and visualized, using chosen security alerts as the label (represented by a distinct color). It could show, hypothetically, that alert 001 occurs predominantly when Mac OS Mojave is combined with a certain range of agent versions.

Complete flow of a clustering analysis, with data prep, class balancing, entity embedding training, and finally clustering and visualization.

Conclusion

Cyber security software must run in hyper diverse environments, with almost unlimited possible configurations that are ever changing. The potential side effect of having the software agent inadvertently crash 3rd party apps leads to unwarranted user suspicion that all crashes are linked to it, when the real cause may be due to any of the co-resident 3rd party apps. Isolating the cause-effect relationship behind a crash requires collecting and analyzing a dataset that is both enormous and high dimensional. The effort starts at the data collection stage, where both the infrastructure and process must be properly set up to generate reliable and trustworthy inputs. Besides simple charting of app-pair correlations, a more advanced level of clustering analysis can be done to illustrate group level behavior among the possible alerts raised by the agent, and how they differ from one machine environment to another.

BigR.io is a US based consulting company with its headquarters in Boston. BigR.io empowers its clients to drive innovation and achieve the "intelligent enterprise" through the use of contextualized data and sophisticated ML capabilities. When it comes to Cyber Security, BigR.io has deep experience in proactive cyber and physical security for equipment and workers, with our excellent expertise in this domain across all phases of the project and the supporting enterprise functions. Thus, if you are looking at a test model or a workable process and are interested to partner with us you can write to us innovation@bigr.io

BigR.io, LLC Harvard Square, One Mifflin Place, Suite 400, Cambridge, MA 02138
(617) 500-5093 | info@bigr.io | www.BigR.io