A Survey of Technical Approaches for Developing, Deploying, and Adopting Visualizations in the Cybersecurity Domain

Robert Gove*
Two Six Labs

ABSTRACT

Members of the visualization cybersecurity research community are doing important work designing and developing new visualization tools to solve important cybersecurity problems, but technical challenges remain in transitioning and deploying visualizations to end users. This abstract compares and contrasts four common technological methods for developing and deploying visualization tools in the cybersecurity domain. The advantages and disadvantages of each are based on the perspectives of software developers, system administrators, and users who will need to develop, deploy, and adopt the visualizations. This abstract identifies gaps in the current methods, and proposes possible solutions for software frameworks to improve support for developing, deploying, and adopting visualizations in cybersecurity environments. The goal is to begin a discussion on the role technology plays in the success of cybersecurity visualizations.

1 INTRODUCTION

Members of the cybersecurity visualization research community are doing important work building visualization tools and prototypes, but the next step is to operationalize the visualization tools, and this can be more or less challenging depending on the frameworks and software architecture available. In particular, certain aspects of the cybersecurity domain impose some restrictions not present when developing and deploying visualizations in other fields. In order to build a successful visualization tool, developers need to be mindful of their available options and the factors that can influence the success of their visualization tool.

This abstract presents a brief survey of four common visualization "frameworks", ranging from BI tools like Tableau to tailor made client-server web applications. (For the purposes of this abstract, the term "framework" is used loosely.) The goal is to begin a discussion about the technical requirements for deploying visualizations in cybersecurity environments. The discussion in this abstract addresses the situation where a cybersecurity team, such as a SOC (Security Operations Center), needs visualization capabilities not supported by their existing tool set. This could be adding support for basic charts, like bar charts or scatterplots, or it might mean creating a highly customized visualization to understand the results of a machine learning algorithm. This survey describes the advantages and disadvantages of each framework from the perspectives of the software developer, the system administrator, and the end user, based on insights gained from applying visualization in large US Government cybersecurity programs. The discussion concludes with suggestions for new frameworks to provide better support for visualization development in cybersecurity environments.

2 VISUALIZATION FRAMEWORKS

This comparison focuses on four common frameworks used for deploying visualization to a cybersecurity environment: Off-the-shelf BI (Business Intelligence) tools, customized BI tools, desktop applications, and client-server web applications. Sections 2.1, 2.2, 2.3, and 2.4 discuss advantages and disadvantages of each of these, and Table 1 presents a summary of the trade offs of these frameworks.

2.1 Off-the-shelf BI Tools

One common method for adding visualization to an environment is to deploy off-the-shelf BI tools. Common examples include Tableau[1] and Spotfire[2]. These types of tools can be deployed in addition to other analysis tools and software, and can be either desktop software or client-server web applications.

Advantages These tools are designed to be highly configurable to work with almost any database, making it easy to adapt to a variety of data sources. BI tools typically provide powerful, polished visualizations and user interfaces that can easily be customized to users' needs, for example by changing color scales or changing which variables are on which chart axis. In some cases, BI tools have some support for collaboration, such as creating a visualization or dashboard and sharing it with others. There is low development overhead, since the functionality was already developed by the software manufacturer. In some cases, the BI tool might already be approved to deploy in the target environment—this is important because it can be difficult to get software approved for deployment when sensitive information is involved, such as at banks, hospitals, or classified Government systems.

Disadvantages Typically BI tools have a fixed set of supported chart types, so they have low extensibility, and no ability to add new visualizations. BI tools also often lack support for persisting meta data, which can be useful for marking data with analysis notes [1]. Although some BI tools support many operating systems, like Kibana[3] and Tableau, some BI tools only support Windows, like the current version of Microsoft Power BI[4].

2.2 Customized BI Tools

Although some BI tools can have restricted extensibility, others like Kibana and Microsoft Power BI come with the ability to write plugins, and some like Superset[5] are open source tools that can be extended. Consequently, they represent a hybrid approach between an off-the-shelf solution and a completely new visualization tool built from the ground up.

Advantages If the BI tool is already approved for deployment, it can potentially be easier to get plugins approved to be installed than entirely new visualizations applications. This can be because there is less code to review, and the plugins might be running in a sandboxed environment that restricts functionality. Plugins can leverage the extensibility of the host BI tool, e.g. for supporting a variety of data sources or ability to change color scales, which can reduce the development cost. Plugins can also be easier to install than an entirely new software systems, but they might need an administrator to install them.

*e-mail: robert.gove@twosixlabs.com

[1] https://www.tableau.com/
[2] https://spotfire.tibco.com/
[3] https://www.elastic.co/products/kibana
[4] https://powerbi.microsoft.com/
[5] http://airbnb.io/projects/superset/

Disadvantages Customized BI tools can potentially have limited design flexibility ability due to restrictions in the API and database queries. Furthermore, plugins also have little to no capacity for a backend, which limits some kinds of functionality like preprocessing the data to speed up the visualization. In some customizable BI tools, such as Kibana, the API is undocumented, which increases development work and can nullify any savings in the development cost. In other cases, the BI tool might not support your target operating system (e.g. Microsoft Power BI does not currently support OS X or Linux operating systems).

2.3 Desktop Applications

New visualization tools can be developed as stand-alone desktop applications. These could be developed in a variety of ways, such as Java applications, Linux packages such as RPM, or frameworks like Electron[6] that allow desktop applications to be developed as client-server web applications.

Advantages Because these tools are often built from the ground up, the possible design space is huge, and developers have few limitations for tailoring the tool to the users' use cases. Desktop applications can be designed to operate offline from a local data source, interface with a networked database, or both.

Disadvantages Although desktop applications can be highly customized by the visualization developer, from the user's perspective customization is limited to the functionality added by the author, e.g. users may not be able to choose color scales unless the developer specifically adds that functionality. Compared to BI tools with sharable dashboards, desktop applications can present difficulty with collaboration and sharing analyses, transformations, and dashboards. Because of security risks from unknown code, it can be difficult to get approval to deploy desktop applications in secure environments. The software developer must plan for all possible different deployment operating systems, which may necessitate using cross-platform software tools such as Electron or Java. Finally, desktop applications must be installed on each user's computer, which means that even if the visualization tool is easy to install, every user who wants it must install it, thereby increasing friction and reducing user adoption rates.

2.4 Client-server Web Applications

New visualization tools can also be developed as client-server web applications, where a backend server and database work in conjunction with a frontend GUI loaded in the user's web browser. Examples include backend and frontend using the MEAN stack (MongoDB[7], Express[8], AngularJS[9], and Node.js[10]), or a framework like R[11] and Shiny[12]. Although web applications have many of the same advantages and disadvantages as desktop applications, there are a few differences, summarized below.

Advantages In contrast to desktop applications, installation is only needed on the server, instead of on every desktop, which can reduce the barriers for users to adopt the tool. This also reduces the need for supporting multiple operating systems (although cross-browser support can become an issue). Web-based tools can provide better support for collaboration, because all the data is stored in a central server, and users can share links to visualizations they created. The caveat is that support must be added by the visualization developers, representing a development cost.

[6]https://electron.atom.io/
[7]https://www.mongodb.com/
[8]https://expressjs.com/
[9]https://angularjs.org/
[10]https://nodejs.org/
[11]https://www.r-project.org/
[12]https://shiny.rstudio.com/

Disadvantages Web applications can be more difficult to install than desktop applications, although using a containerization framework like Docker can help simplify installation. Web-based tools also cannot be used if the user is offline and not connected to the network.

3 Discussion

Table 1 summarizes the four frameworks and their advantages and disadvantages.

As we can see, there are numerous trade offs and gaps in the available technology. This discussion can help visualization developers carefully weigh their options and decide on the best technical approach to solving the visualization needs of their cybersecurity environment. This discussion can also inform the development of future visualization frameworks. For example, by examining Table 1, we see that customized BI tools partially support almost all of the features. By carefully designing a new, customizable BI framework based on the requirements of operationalizing visualization for cybersecurity, it may be possible to provide visualization developers with the support they need to develop powerful visualizations while also minimizing the effort to design, develop, and deploy them.

Table 1: Summary of each framework. **OTS BI** is off-the-shelf BI tools, **Cust. BI** is customized BI tools, **Desktop** is desktop applications, and **Web** is client-server web applications. An empty cell indicates no support, a half filled circle ◐ indicates partial support, and a filled circle ● indicates full support.

Feature	OTS BI	Cust. BI	Desktop	Web
Configurable data sources	●	◐	◐	◐
Configurable visualizations	●	◐	◐	◐
Design flexibility		◐	●	●
Data persistence			●	●
Collaboration support	◐	◐	◐	●
User OS support	◐	◐	◐	●
Easy installation	●	◐	◐	◐
Low development cost	●	◐		
Low approval overhead	◐	◐		

Acknowledgments

This work was funded by the Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement 'A' (Approved for Public Release, Distribution Unlimited)

References

[1] R. Gove, J. Saxe, S. Gold, A. Long, and G. Bergamo. Seem: A scalable visualization for comparing multiple large sets of attributes for malware analysis. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, VizSec '14, pages 72–79, New York, NY, USA, 2014. ACM.