Validating Network Security with Predictive Analytics:

A Design Guide to Bridge Stochastic Modeling Into A Computationally Intelligent Dashboard

A Research Thesis Presented To:

The Information Design & Technology Program

State University of New York at

Polytechnic Institute

By: Christopher R. Galavotti

May 2019

Certificate of Approval

SUNY Polytechnic Institute

Department of Information Design and Technology

Approved and recommended for acceptance as a thesis in partial fulfillment of the requirements for the degree of Master of Science in Information Design and Technology.

_____ Date

_____

Dr. Russell Kahn

Research Thesis Advisor

_____

Dr. Kathryn Stam

Second Reader

Abstract

Network posture has historically relied on traditional and reactionary methods for protection. These methods most commonly consist of network segmentation, intrusion detection systems, intrusion prevention systems, and signature-based detections. However, these traditional security platforms have proven to be an inadequate deterrent to the complex threat matrix that we currently find ourselves in. It is only through computational intelligence that we can truly identify potential intrusion areas and network abnormalities. This study presents a path forward for industry professionals on how to implement this computational approach into their network security platforms, particularly through stochastic modeling and simulation. Acknowledging the complex nature of this approach, a human-centered design methodology is also outlined on how to integrate this science into the enterprise via a predictive analytical dashboard.

Keywords

Network Security, Stochastic Modeling, Attack Graphs, Predictive Analytics, Exploitability Benefits, Markovian Chains, Computationally Intelligent Systems, Principles of Design, Human-Centered Design Theory, Model Simulation and Acceleration, Software Development Process

Table of Contents

Introduction

Various research and new age methodologies all point to why it's necessary to complement traditional defense-in-depth network security with predictive analytics (Buczak and Guven, 2016, pg 1153). Traditional security methods have proven time and time again to be reactionary in nature and lack the predictability required to prevent incident and intrusion. As many in the industry already know, every external facing host is a possible intrusion point that can only be mitigated to a certain degree by conventional means. Mitigation techniques for these endpoints are often emphasized through inadequate physical and logical network barriers. This study however, strives to break that conventional mindset by providing a guide on how to implement an additional layer of predictive analytics for enterprise integrity and sovereignty.

In particular, various stochastic models are presented to show just how effective they can be for data loss prevention and overall security. In fact, their origin along with other network statistical modeling and simulations can be traced back to the DoD in the late 1980's (Pokhrel and Tsokos, 2017, pg 92). It was there that the graphical representation and modeling of attack simulations provided revolutionary insight into the defense-in-depth strategies we currently use today.

These stochastic methods have proven effective within networked environments due to their allowance for random variations. It is this permission of random variation that allows for a more accurate estimate of a potential distribution or outcome. Moreover, only this type of modeling allows for a probabilistic vision that has the varying complexity issues found in networks and their threats. Simply put, by building the appropriate stochastic model and understanding the correlation effects of network vulnerabilities, it is in fact possible to predict future attacks. This insight then in turn allows for the distribution and optimization of resources accordingly to prevent those attacks.

It should be noted that building network models, graphical representations, and simulating results to predict unconfirmed behavior is nothing new; however, the challenge has always been on how to integrate, automate, and present that level of computational intelligence into a useable monitoring tool for overall network security. From personal experience, more often than not, the failure of technology is not due to its effectiveness or applicability, but rather it fails to meet certain design conceptuality standards. This is perhaps an inevitability given the nature of the right-brain/left-brain dominance theory. What this guide strives to stem is the gap between the two dominances, and help ensure that this type of statistical modeling can be utilized effectively and with purpose.

Technical Terminologies

Attack Graph - a visual or logical representation of the various potential paths an unauthorized person can use to compromise a system, using one or more vulnerabilities (Abraham and Nair, 2015, pg 3).

Canonical Form - is a way to represent a mathematical object or entity in its most standard form. In most cases, this form creates a representation for each unique identifier of that object (Abraham and Nair, 2015, pg 7).

Common Vulnerability Scoring System (CVSS) - a score that is representative of a computer systems overall security vulnerability. Scores are formula based and range from 0 to 10, 10 being the most severe exploit (Pokhrel and Tsokos, 2017, pg 94).

Exploitability Benefit - is a mathematical way to express the value of obtaining one vulnerability or networked node from another. This is significant for attackers because it determines the most valuable path of intrusion (Pokhrel and Tsokos, 2017, pg 97).

A Markov Chain - relates to a form of stochastic model where an outline of events are succession based from previous events. In probability theory, it is a mathematical transition from one to another provided by certain probabilistic rules (Pokhrel and Tsokos, 2017, pg 94).

Predictive Analytics - is the data cultivation of various subsets used to predict future outcomes. These data sets can be cultivated through various mathematical models, statistical techniques, machine learning, or even soft computing (Abraham and Nair, 2015, pg 2). Data sets are used to highlight potential outcomes and trends given data patterns.

RTVM – a Requirements Traceability and Verification Matrix (RTVM) is commonly used in software or systems engineering to ensure all requirements are captured and verified (Gechman, 2011, 2-64).

Literature Review on Stochastic Modeling

There is very little debate on the digital conflict and complex computing environment that we currently find ourselves in. However, by utilizing stochastic modeling, a forecast of overall network security can be

made. This predictive forecast allows for the pinpointing of vulnerabilities and their possible exploitations in a very accurate way.

Such stochastic design models point precisely to this indicated success. One such model was created using the Markovian process in conjunction with the Common Vulnerability Scoring System (CVSS) framework. This model was created by N.R. Pokhrel and C.P. Tsokos from the University of South Florida. Their model, as they highlight in their study, can be used to categorize significant network nodes where intruders are most likely to focus on (Pokhrel and Tsokos, 2017, pg 93). Once this is identified, system prioritization and designated patching can occur. They also cite a previous model that can actually help predict path length and evaluate various vulnerabilities to allow for the correct amount of system resources to be allocated. They further expanded upon that model so that it's now possible to predict vulnerability assessments network wide through the exploitability sub-score and impact sub-score method (Pokhrel and Tsokos, 2017, pg 94).

Another significant study involving statistical models works to create identifiers to critical elements in a system via an attack graph. This identification allows for the hardening of defenses based off predictive indications that an intrusion will occur within those particular parameters. What is most significant about this study by Abraham and Nair is that their stochastic model factors in a temporal account related to vulnerabilities that can change over time. This temporal factor they consider is the time dependent covariate, specific to the age of a given vulnerability. By taking this into account, particularly in terms of the availability of exploits and their associated patches, Abraham and Nair created a platform capable of predicting the future security state of any given network, at any given time (Abraham and Nair, 2015, pg 2).

A Stochastic Predictive Model

Assuming a conventional network topology, N.R. Pokhrel and C.P. Tsokos construct a host access graph and a one state probability matrix utilizing the exploitability sub-score and impact sub-score. This sub-score utilization allows for each network node, or in this case, each vulnerability point, to be ranked and evaluated. This evaluation is critical because it then allows for targeted threat mitigation. It's also important to note, as many have before, that standard attack graphs are often ineffective due to their inability to computationally evaluate complex and numerous threats.

Pokhrel and Tsokos highlight this, and to address it they recreated an attack graph that is essentially a two-layer graph. The upper layer being the host access graph, and the lower construct being comprised of the host pair attack graphs. It is this type of attack graph that they built their stochastic model from. To be conceptually viable, it is important to note that of this recreated attack graph, it is the lower level that describes the potential vulnerability scenarios between a given host, and it is the upper level that shows just how the network access between those hosts occur (Pokhrel and Tsokos, 2017, pg 95).

Given the above, utilizing a Markov chain is an essential piece to this model. This mathematical progression of events is ideal for network intrusion because as with a Markov chain, each potential event is based on the state obtained from the one before. They start by utilizing this chain, letting S be the potential number of states, specifically related to the number of nodes in the host access graph, and X being the random variables, represented as:

$$P\left[X_{n+1}=y \middle| X_0=x_0, X_1=x_1, \cdots, X_n=x_n\right]=P\left[X_{n+1}=y \middle| X_n=x_n\right]$$

One important caveat in their proposed model suggests that the node transitions, or the Markov walkthrough, is memoryless. Meaning that once the attacker has gained access to one node, every step before that has been forgotten. All that is realized is the path forward to the goal node. This is where the exploitability sub-score and impact sub-score method comes into play. This is mathematical represented with the “bliss factor” (individual skill of the attacker) as:

$$P(x,y)=P\left[X_{n+1}=y \middle| X_n=x\right], \text{ for all } n.$$

Finally, given the analytical framework they present (the Markovian process in conjunction with Common Vulnerability Scoring System (CVSS) framework), and the host access graph mentioned, their model representation is displayed as:

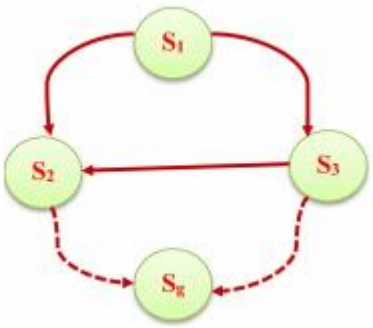

Figure 1: Host Access Graph (Pokhrel and Tsokos, 2017, pg 96)

This model also assumes:

$S_i, i = 1, 2, 3, \cdots, g$ are host nodes and S_g is a goal node

To preface, each node is modeled after a host in the access graph, and therefore the number of nodes is equivalent to the number of hosts in the network. The solid line running from S1 to host 2 represents the access available on 2 S from S1. The dashed lines from host 2 S to host g S, highlights that there are in fact other nodes present and that the same explanation is applicable to other nodes (Pokhrel and Tsokos, 2017, pg 96).

This model goes further on to account for the possibility of node selection when there are multiple nodes to reach the end goal, and just how the walkthrough selection process is determined. To account for this, they create an exploitability benefit which can be expressed by:

$$\mathbf{ExploitabilityBenefit} = f\left(\mathbf{Exploitability}, \mathbf{Impact}\right)$$

Using this exploitability benefit to mathematically value one node from another, the model can now be represented.

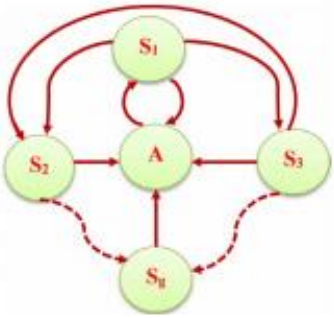

Figure 2: Updated Access Graph (Pokhrel and Tsokos, 2017, pg 97)

Further assumptions are made for this model regarding the risk based on rankings, and the physical/logical network environment that this model fits into for their study. Their overall model creation, while only covered here briefly and not in its entirety, does seem to suggest that it can help

network administrators determine their overall security risk for any given network. Pokhrel and Tsokos's stochastic framework outlined above, if nothing else, provides proof of a rather unique and perhaps effective approach to creating a new security evaluation. Utilizing a Markov walkthrough process in conjunction with a CVSS framework does prove to be a great analytical tool that has the potential to be scaled to match the most complex network environments of the future (Pokhrel and Tsokos, 2017. Pg 97).

Attack Graphs with Temporal Factors

As mentioned in the stochastic predictive model above, creating and validating attack graphs can be an incredibly useful tool for threat analysis within network security. In the simplest terms, an attack graph is a rudimentary chain of vulnerabilities. Most data breaches or perhaps even penetration testing starts at one vulnerability and migrates up the chain, so to speak. This accumulation of vulnerabilities, or chain, creates what is called an attack path, and mapping out this path leads you to develop an attack graph.

This platform is based on a typical network security analytical framework outlined below as well as utilizing the traditional scoring method of CVSS that was previously outlined.

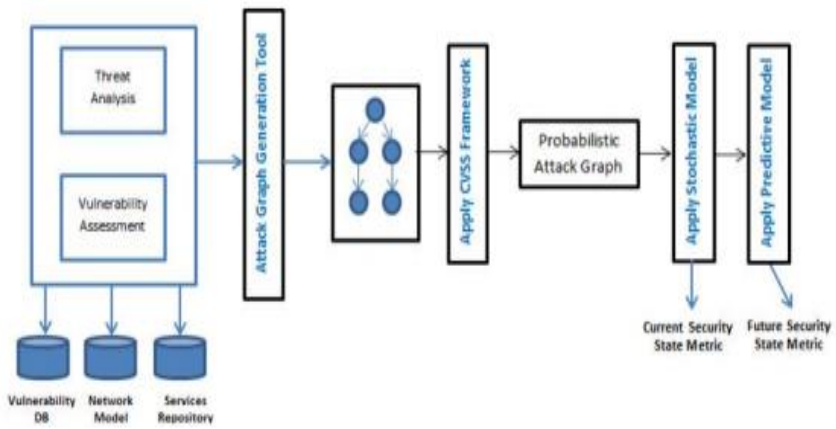

Figure 3: Network Analytics Framework (Abraham and Nair, 2015, pg 5)

This model is developed on a four-layer construct where each layer builds upon the previous one. The first layer is the attack graph model itself, created by inputting various network topologies and then running a series of simulated breaches on each vulnerability. The second layer is the above mentioned CVSS construct. Layer three represents the stochastic input where processes are injected over the graph to

outline the relationship between the various potential vulnerabilities. For this specific model, Abraham and Nair use the same Markov chain processes as outlined above for the impact assessment. Finally, the fourth layer represents potential vulnerability development. This is perhaps the most significant factor in their model because it takes into account the variable of the ever-changing threat landscape. These temporal properties of each vulnerability are what allows this model to evolve along with the threats.

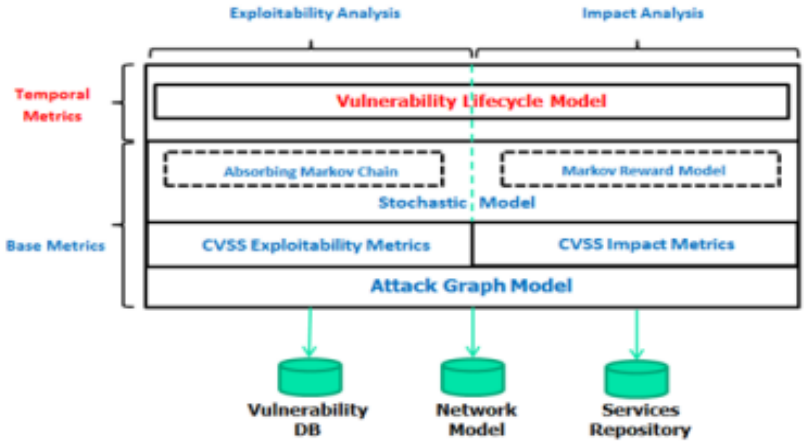

Figure 4: High Level Model Architecture (Abraham and Nair, 2015, pg 6)

In the model below, it is assumed that the state si of a Markov chain is to be considered absorbing, and therefore it is not possible to leave (i.e., pii = 1). Also note a state not to be considered absorbing is classified as a transient state (Abraham and Nair, 2015). With respect to their model, state 4 is to be considered absorbing with 1 associated to its probability, and the rest of states (1, 2, and 3) are to be considered transient. A visual representation of the model is:

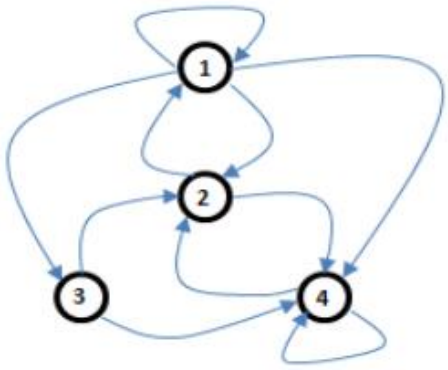

Figure 5: Absorbing Model (Abraham and Nair, 2015, pg 7).

To further this study, Abraham and Nair take this model and expand on it by creating a formula for computationally evaluating the probabilities of the Markov chain via normalizing the CVSS vulnerability numbers. First, by understanding the vulnerability area that will be targeted, metrics can be devised for what it will ultimately take to meet that goal.

The computational elements of the model are outlined below:

Utilizing the CVSS framework for calculating scores, they express this using the base exploitability score e (v) to replicate the complexness of exploiting the vulnerability (v). The access vector (AV), access complexity (AC), and authentication factors as (Au) (Abraham and Nair, 2015, pg 8):

$$e\ v\ = 20 \times AV \times AC \times Au$$

Then by calculating the base exploitability score and the temporal weight, the effective temporal exploitability score is as follows

$$e\ v_t\ = temporal\ weight \times e\ v$$

The transition matrix for an absorbing Markov chain has the following canonical form:

$$P = \begin{matrix} Q & R \\ 0 & I \end{matrix}$$

The transition probability matrix P of the Markov chain was estimated using the formula:

$$p\ i,j\ = \frac{e(v_j)}{\sum_{k=1}^{n} e(v_k)}$$

In an absorbing Markov chain, the probability that the chain will be absorbed will always have a factor of 1. So therefore

$$Q^n \to 0 \; as \; n \to \propto$$

In order to determine the market availability of a vulnerability exploit, a Pareto distribution can be used and expressed as:

$$F\ t\ = 1 - \left(\frac{k}{t}\right)^{a}$$

The expected path length (EPL) metric is represented by:

$$t = Nc \; where \; for \; all \; j \; c_j \; is \; 1$$

Calculating the impact associated with a given vulnerability can therefore be measured by:

$$Impact\ v\ = 10.41 * (1 - \ 1 - C\ *\ 1 - I\ * (1 - A)$$

Abraham and Nair then simulated attacks that were occurring on four systems within a typical network topology outlined in the below image.

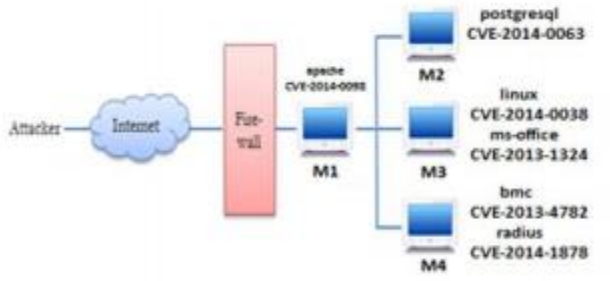

Figure 6: Simulation Topology (Abraham and Nair, 2015, pg 11)

Given a random generated set of environmental vulnerability data, an attack graph was generated, toolkits were created, and an absorbing Markov chain was introduced.

Service Name	CVE-ID	Exploitability Subscore	Host	Disclosure Date
apache	CVE-2014-0098	10.0	M1	03/18/2014
postgresql	CVE-2014-0063	7.9	M2	02/17/2014
linux	CVE-2014-0038	3.4	M3	02/06/2014
ms-office	CVE-2013-1324	8.6	M3	11/13/2013
bmc	CVE-2013-4782	10	M4	07/08/2013
radius	CVE-2014-1878	10	M4	02/28/2014

Figure 7: Threat Data (Abraham and Nair, 2015, pg 12)

To put their model to the test, various simulations were conducted with modeling of attacks that simulated over 2000 more vector variations. These attack instances were done over the attack graph and were connected to the relevant probability location of the nodes. Depicted within their simulation results show the various colors and how they correlate to trend forecasts over a given period in time. Each simulation that was created used the transition probability of the absorbing Markov chain to jump from one node to another until the final goal was reached.

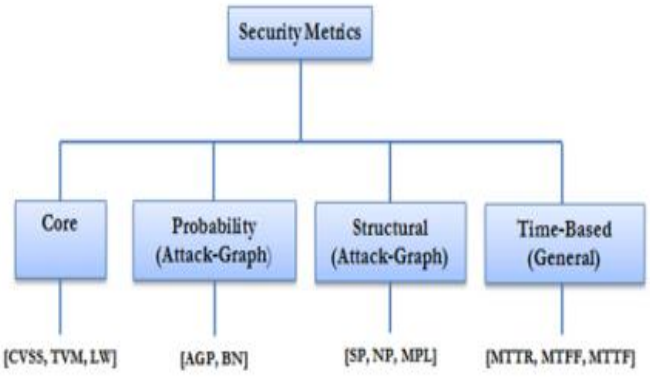

Figure 8: Metric Results (Abraham and Nair, 2015, pg 3)

Designing the Predictive Analytical Dashboard

Given the success these models and simulations have empirically outlined, the question now becomes not whether to utilize them for predictive network analysis or not, but rather how to integrate them into practical use for an enterprise. Unfortunately, their practicality becomes vague and abstract when they are detached from the frontline security professionals that need them most.

The solution to this is creating a software-based analytical dashboard that can be available and operational at the localized networked environment. The rest of this study serves as a guidebook on how to efficiently and effectively design a human-centered, software-based solution that implements the stochastic modeling techniques outlined above.

In order to be effective, this type of dashboard needs to utilize not only an applicable software engineering methodology, but also a Human-Centered Design (HCD) one. Within most enterprises, merging this science into a predictive analytical dashboard that is focused on the security and network administrators is the critical link that has been missing for successful implementations. This has been historically overlooked during the software development phase. However, this critical omission can be mitigated by simply creating and maintaining a Concept of Operations (CONOPS) and Software Development Plan (SDP) that is based off the HCD theory.

Human-Centered Information Design

The application of design principles within the software engineering community has been an interesting subject of study for many years, especially as it relates to the development of data analytics through visualization. This art is perhaps one of the most misunderstand and underdeveloped practices in the information technology sector. It should be well known that in order to create a useful enterprise software solution, appropriate design theory and methodologies need to exist at the onset. In this particular case, stochastic modeling requires such complex computationally intelligent elements, an HCD theory will create an effective software tool for true network protection and implementation.

As highlighted within Cairo, “the art and science of preparing information is so that it can be used by human beings with efficiency and effectiveness, and that the goal of that information is that it can be navigated effortlessly” (Cairo, 2001, pg 18). There have been numerous other studies that highlight this being the precise reason why many information technologies fail. Leading research in this field by Jacobson also highlights just how paramount it is to create technologies that are focused around the human design (Jacobson, 1999, pg 66).

Given this research, it's clear that when designing or overseeing this type of software for an enterprise, one must be cognizant to implement principles focused around the HCD construct. As outlined by Jacobson, those involved in any form of system design need to be competent in the design areas that focus on a coherence of meaning, inclusiveness to the point where the user feels invited in, malleability so it is possible to mold situations, all around engagement for the user, and a certain level of responsiveness and ownership (Jacobson, 1999, pg 68). To help with this, utilizing a HCD process model may prove beneficial. Consistently referring back to such a model throughout the software lifecycle is the best approach. Below is a proven and tested process model published by JMIR Human Factors:

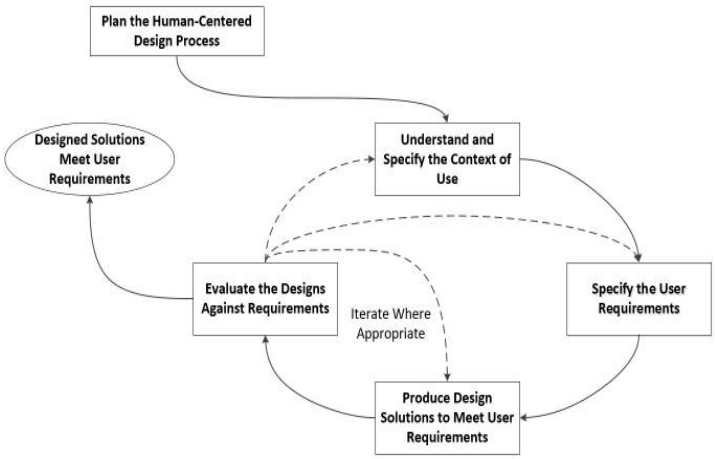

Figure 9: A Human-Centered Design Process Map (JMIR Human Factors, 2017, pg 4)

As shown above, this process map looks much like it would for a software engineering process map. Planning the HCD process is much like developing a CONOPS as well, and you will notice that much of the elements in the process relate to many elements that are found in most standard Software Development Plans (SDP's). Even though they are similar, what most enterprises fail to do is merge the HCD methodology within their software engineering process.

Creating a Human-Centered CONOPS

When operating within a standard software or systems engineering framework, one of the most central steps for successfully fielding any product is creating a CONOPS. A CONOPS is a living document that

can and should be used throughout design creation and implementation - all the way through to the later phases of the software lifecycle process. Its purpose is to outline the quantitative and qualitative system requirements of a proposed system or software.

As indicated above, the integration of stochastic based software into an enterprise must have a CONOPS that utilizes the HCD framework. Just as the purpose of a CONOPS is to describe the overall characteristics of the planned system or software, the HCD model consistently focuses on the user, their needs, environment, workflow tasks, and services. It is quite easy to see how these processes must be done jointly, as this is the only method that allows for HCD solutions to be implemented before it's too late.

In order to get the predictive network analytics most enterprises need, we know the software's graphical user interface needs to be able to scan all critical network elements and calculate predictive indicators that future intrusion will occur at those elements. Once these elements are identified, mitigation options must be calculated and remediation capability done through the interface. However, again, what most designers will miss is that by implementing a HCD methodology into that overall framework, they will increase the usability and practicality of the dashboard software. One simple and easy way to ensure your CONOPS is focused on this HCD construct is to always reference the four defined activity phases throughout development. These phases include:

(1) Identify the user and specify the context of use upfront;
(2) Specify the user requirements before development begins;
(3) Produce design solutions in accordance with those requirements, and
(4) Evaluate design solutions against said requirements (JMIR Human Factors, 2017, pg 7).

After these high-level plans are formalized, a SDP must be created and implemented to see the CONOPS and your new HCD requirements through to completion.

Creating a Human-Centered Software Development Plan

One of the best software development guidebooks on the market today was piloted by the Air Force Space Command out of El Segundo, CA. This is an industry leading guide that all enterprises need to adhere to in order to effectively field a modeling-based software program within their organization. It outlines the critical importance of a correctly compiled SDP. Firstly, it points to how pivotal the

integrated product team plays on your software creation. As Gechman emphasizes, establishing an effective integrated product team is the most significant ingredient to a successful software development program (Gechman, 2011). Establishing this team early on in the process, much like with your CONOPS, allows for a HCD focus to occur. This integrated product team is critical to help determine, integrate, and verify your HCD requirements because they are the principle stakeholders responsible for delivering the

Next, given the complexity predictive modeling requires on enterprise level software, it's even more important to understand the various activities that are required within the software lifecycle process. The figure below outlines the true breadth of what it takes for a successful software implementation to occur.

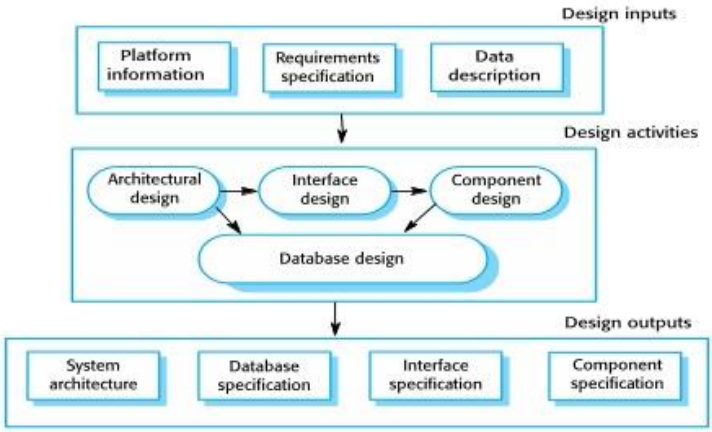

Figure 10: Model of the Design Process (Software Engineering 10, 2015, pg 6)

When evaluating the above, it is critical to be thinking of each domain (management, system, support, and software) in terms of this HCD methodology. Take for example, one of the biggest operational burdens within this process activity chart is software documentation and requirements analysis. For this, most software designers utilize a Requirements Traceability and Verification Matrix or RTVM. This is done so correlations can be tracked and later verified in a consistent and traceable framework. In the most standard terms, a RTVM is used to determine that specific functional and nonfunctional requirements are applied and tested for a given software.

An interesting point to note is that this same methodology can be applied to ensure that you are capturing the necessary HCD requirements as well. In fact, HCD theory has roots in the field of requirements engineering because it also seeks to document the user requirements and how they are being met by the design at each stage of development (JMIR Human Factors, 2017, pg 5).

You will not find any product manager or software engineer that will disagree with how important it is for enterprises to utilize some form of RTVM for system requirements. What you rarely find however, is the understanding that before they develop specific functional and nonfunctional requirements within this RTVM, they need to meet the six HCD requirements. These requirements are not in contrast to standard definition requirements, but complementary to them. In fact, the specific functional and nonfunctional requirements should be assumed based off these HCD ones. The requirements are outlined by ISO 9241-210, which are international industry standards recognized worldwide. They are outlined as:

(1) Design requirements should be based on an explicit understanding of users, tasks, and environments;
(2) Users should be involved throughout the design and development;
(3) The design is driven and refined by user-centered evaluation;
(4) The process is iterative;
(5) The design addresses the whole user experience and not just part of it;
(6) The design team includes multidisciplinary skills and perspectives (JMIR Human Factors, 2017, pg 7).

Without a SDP that is heavily grounded in these HCD requirements, the enterprise level software you are wanting to create might certainly be functional and incorporate the required statistical predictive modeling, but it will undoubtedly fail to be of practical use for the practitioner.

Other Human-Centered Principles

By now it should be evident that in order to successfully implement a predictive network security software solution for your enterprise, a HCD approach is required. Aside from tailoring your CONOPS and SDP to this methodology, there are two other final design principles that one should capture for this type of software-defined dashboard. Although there are dozens of design principles, the notion of application distilling and color theory are the ones most likely to be overlooked, and therefore deserve further explanation.

Application Distilling

Perhaps one of the most important practices when implementing a complex predictive analytical dashboard is presented by Daniel O'Sullivan in the “Six Steps to Designing Better Dashboards.” Here he introduces this notion of application distilling. It’s a simplest enough concept, but it is absolutely critical.

As O'Sullivan highlights, one of the biggest mistakes software developers make is that they don't adequately reduce their application down to only its most important functions (O'Sullivan, 2016, pg 1). He therefore provides a mini guide on a proven method of how to identify just the key functions that are needed for an analytical dashboard based off mathematical modeling.

The 5 Step Guide to Application Distilling by Daniel O'Sullivan:

Step 1: Take a large whiteboard and write 3 words across the top: Features, Navigations, and Metric.

Step 2: Give the stakeholders of the software a marker and 3 sticky notepads in different colors. Have them all list out everything the application does in each of those categories, and then place each one on the board under its respective heading.

Step 3: Draw a large rectangle on the board to represent the dashboard with a section across the top or left-hand side for the navigation.

Step 4: Place the links in the navigation area and the features and metrics in the main area. Note: this process may take a considerable amount of time with lots of back and forth. As the information designers, they should challenge the business case for every sticky note placed on the board.

Step 5: Refine the final sticky notes left standing on the board to their basic elements. For example, if they have a file storage widget, choose which secondary actions can be performed (copy, download, share, delete) (O'Sullivan, 2016, pg 1).

Color Theory

Another important characteristic of your software-based dashboard will be your choice of color palette. If you're in the business of dashboard design, it is strongly recommend understanding the concepts behind Alberto Cairo's visualization wheel. Such things as choosing complementary colors to highlight data, understanding some features need to be represented on the more intelligible and shallower pole of the wheel and some more on the complex or deeper side of the spectrum is paramount.

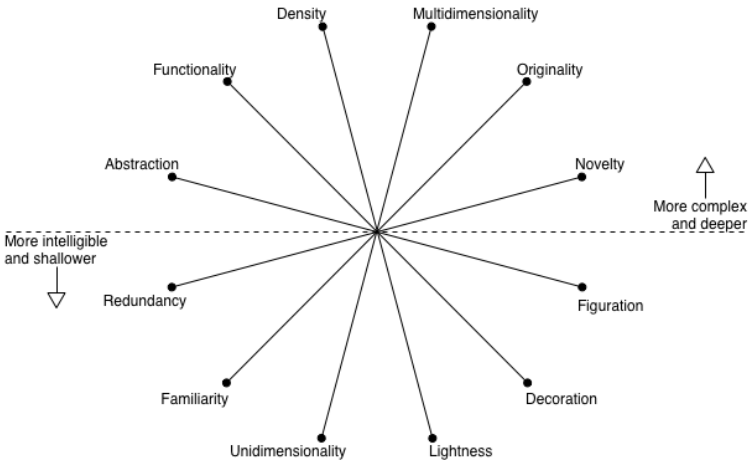

Figure 11: Alberto Cairo's Visualization Wheel (Cairo, 2013, pg 50)

Also, when designing the overall dashboard layout, think of these guiding principles outlined by Cairo.

1) Think of your dashboard console in layers. Start by offering a summary of the data. This is generally done on the home toggle. Let this layer serve as the entry point into the software, clueing the users into the other areas of access.

2) Beneath that layer, think of all the possible navigation panes as inner layers of information. No need to have them all mapped out, but make editorial decisions based on the HCD framework presented earlier.

3) Be sure to think of how the navigations or inner layers should be structured. In some cases, the structure will be linear or logical, in others, you can organize the navigation so that readers can explore as they wish (Cairo, 2013, 75).

Finally, I always liked the practice of sticking to the stoplight analogy when it comes to analytical dashboards. As O'Sullivan suggests, "unless you have very specific reasons not to, be sure to use greens and blues for metrics that are tracking well and reds and oranges for those that are not" (O'Sullivan, 2016, pg 1).

Conclusion

Unfortunately, there is no debate that network security is undergoing a transformational period where traditional platforms are no longer adequate as the only layer of defense. A complementary posture that includes more predictive modeling and simulation is what is truly needed to stay ahead of the complex threat landscape that the industry finds itself in.

With its allowability for random variation and migration through Markovian chains, stochastic modeling is perhaps the most useful way for insight into network vulnerabilities. Two specific stochastic models were presented in this study, one by N.R. Pokhrel and the other by Abraham and Nair that demonstrate empirical evidence that these models can in fact predict future attacks with a high degree of certainty or probability.

As outlined, this predictive analytical technique was presented as a guide for industry leaders to motivate its implementation into enterprises. Well aware of the abstractness and applicability issues of the science for the everyday security professional and enterprise stakeholders, the practicality of a computationally designed dashboard was presented. To help alleviate implementation concerns, several information design techniques were presented to be included within a HCD construct. Special attention was given on how an enterprise should develop a HCD CONOPS, SDP, and how to implement a well-rounded software-based solution focusing on industry leading design techniques. This industry guide was all in the hopes that enterprises realize that this modeling technique is invaluable, needed for the future, and can in fact be feasibly bridged into their enterprise.

Bibliography

Abraham, Subil and Nair, Suku. (2015) A Predictive Framework for Cyber Security Analytics using Attack Graphs. International Journal of Computer Networks & Communications (IJCNC), 1-17.

Alessi, Stephen M., and Stanley R. Trollip. (2001) Multimedia for Learning: Methods and Development. 3rd ed. Boston: Allyn and Bacon.

Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials 18(2), 1153-1176.

Cairo, A. (2013). The Functional Art: An Introduction to Information Graphics and Visualization. New Riders.

Gechman, Marvin. (2011). The Elements of an Effective Software Development Plan – Software Development Process Guidebook. Aerospace ATR 2011(8404)-11.

Harte R, Glynn L, Rodríguez-Molinero A, Baker PM, Scharf T, Quinlan LR, ÓLaighin G. A Human-Centered Design Methodology to Enhance the Usability, Human Factors, and User Experience of Connected Health Systems: A Three-Phase Methodology.

JMIR Human Factors. 2017 Mar 16;4 (1).

Jacobson, Robert. (1999). Information Design. MIT Press. Cambridge, Massachusetts.

Kemmerer, R.A. and Vigna, G. (2002) Intrusion Detection: A Brief History and Overview. IEEE Journals and Magazines, 35, 27-30.

Mehta, V., Bartzis, C., Zhu, H., Clarke, E. and Wing, J. (2006) Ranking Attack Graphs. International Workshop on Recent Advances in Intrusion Detection, Hamburg, 20-22 September 2006, 127-144.

O'Sullivan, Daniel. (2016) Inside Design. 6 Steps to Designing Better Dashboards. InVision.

Pokhrel, N.R. and Tsokos C.P. (2017) Cybersecurity: A Stochastic Predictive Model to Determine Overall Network Security Risk Using Markovian Process. Journal of Information Security, 8, 91-105.