The Impact of Known Vulnerabilities on Layered Solutions

A Timeline of Layered Solution Vulnerability

Robert L. Haverkos

Daniel J. Sokoler

Purdue University

*Produced as part of the INSuRE Project at Purdue University, Spring 2016*

*Professors*

*Dr. Dark*

*Dr. Springer*

*Technical Directors (Industry Sponsors)*

*Trent Pitsenbarger*

The Impact of Known Vulnerabilities on Layered Solutions

6 May 2016

Robert Haverkos - rhaverko@purdue.edu

Daniel Sokoler - dsokoler@purdue.edu

Purdue University

Evaluating points in time where, given known vulnerabilities for a layered security solution, an attacker could have feasibly broken through both layers.

**Keywords**: Layered Solutions, CSfC, Systems Security, Information Assurance

**1-3 Sentence Description of the Project:** This project focused on the observation and analysis of past known vulnerabilities across two mechanisms in a layered solution. Given those known vulnerabilities, and patch times generated by statistical models, we simulated the performance of such a mechanism. An attempt was also made to determine if any of the vulnerabilities in the individual mechanisms are such that they allow for a breach in the security of that mechanism to persist beyond the one week window and allow for an attacker to bypass the layered solution even when the vulnerability windows for the layers do not align.

**Responsibilities:**

* Shared: Literature Review, Sponsor Relations, Data Collection, Analysis, Publication of Deliverables
* Robert Haverkos: Research and analysis of vulnerability and patch publishing trends. Research and analysis of the implications of specific types of vulnerabilities in security mechanisms to layered solutions.
* Daniel Sokoler: Research and development of data harvesting mechanisms to pool relevant data regarding vulnerabilities and patches. Research and analysis of the implications of specific types of vulnerabilities in security mechanisms to layered solutions.

*NOTE: Responsibilities subject to change and amendment throughout the development of the project based on the current needs of the research team*

**Budget:** $17,372

**Deliverables:**

* Project Proposal
* Final Report
* Poster
* Presentation of Project Outcomes
* Timeline
* Scripts used for gathering and analysis

**Executive Summary**

**The Impact of Known Vulnerabilities on Layered Solutions**

May 6th 2016

Robert Haverkos and Daniel Sokoler, Purdue University

*Layered Solutions, CSfC, Systems Security, Information Assurance*

Layered security solutions, in the context of this project, are packages of security measures intended to protect some aspect of an organization's information technology assets. It is a specific case in the larger context of security in depth. To clarify, a traditional view of having multiple layers in a security in depth setting would be to put a locked safe inside of a locked room. This creates a situation where any unauthorized party would have to compromise both the lock on the door and the lock on the safe to gain access to its contents, thereby creating an additional layer of security. The layered solutions that this study is concerned with relate to more to putting a safe inside of a safe, that is, two nearly identical security measures to provide redundancy. These individual measures are referred to here as security mechanisms.

To gauge the assurance of a hypothetical layered security solution consisting of two redundant mechanisms, we will, through observation and analysis of published vulnerabilities for these two mechanisms, identify windows of opportunity where the solution as a whole could have feasibly been compromised..

The first step is the selection of the two mechanisms to use as a test case. In collaboration with our technical directors, we decided that these mechanisms need to be commonly used, have a wide range of information about past known vulnerabilities, and be widely used in their field of protection. Choosing two obscure and rarely used mechanisms would have a much smaller benefit than choosing two well known and widely used mechanisms.

Once the mechanisms have been chosen vulnerabilities need to be gathered and analyzed. The vulnerabilities will be gathered from one of the major vulnerability databases such as the National Vulnerability Database. The analysis consists of two main components. The first component looks for any points in time where a potential attacker could have compromised both mechanisms. This requires each mechanism to individually have a vulnerability severe enough to compromise that layer’s security. The second looks for any vulnerabilities in the first mechanism that are severe enough where an attacker could have compromised the security of that mechanism and then, even after that vulnerability was patched, wait, hidden, in between the two mechanisms until a vulnerability surfaces allowing that attacker to compromise the second layer surfaces.

This gathering and analysis of known vulnerabilities will allow both consumers and producers of these layered solutions to better gauge their effectiveness. Consumers will have the ability to better judge these solution’s effectiveness as well as take stock of any situation in which their solution may have been vulnerable. Producers will have in depth statistics as to their products and can better tailor solutions to fit various situations.

**Motivation**

Layered Solutions, or at least the term as it is being applied in the context of this problem, is a particular property of the Commercial Solutions for Classified program (CSfC) run by the National Security Agency. The program aims to provide an alternative to expensive Government off-the-shelf (GOTS) solutions in information assurance needs involving classified information. [1] In addition to being expensive, GOTS solutions take a long time to develop and test in house which, among other things, makes them slow to market. Commercial solutions have the benefit of a larger market and a larger, more diverse development community making them both cheaper, and quicker to respond to the ever changing world that we face in the discipline of information technology. However, some level of quality control and assurance value is lost in these alternatives. [2] To overcome this degradation in assurance, the idea of creating a layered solution has been proposed. [2] The theory behind this approach is that layering two or more security mechanisms which provide the same security attribute provides a level of redundancy if a vulnerability is discovered in one layer. [2]

While adding additional layers of security to cover a particular threat vector adds a level of redundancy to the system as a whole, it does not necessarily make any of the individual mechanisms, or “layers”, more secure. It is possible, if not likely, that one of these mechanisms will have an open and published vulnerability at some point during the lifecycle of the solution, this is the notion that underpins the idea of a layered solution so that there is still a working mechanism when this occurs. Even within the context of the layered solution, there is the possibility that all of the layers will be vulnerable at the same time, rendering the built in redundancy ineffective. The broader question that this project is attempting to address is how often this alignment of vulnerabilities occurs. Having a better understanding of how often this phenomena occurs will help to better understand the assurance gains of adding layers to a solution and allow for more effective cost benefit analysis when evaluation potential security solutions. Today’s world is information driven. Being able to understand when and how an attacker could have penetrated a system perceived to be secure will allow for a much clearer picture of compromising situations.

This study will attempt to display windows of opportunity where the solution as a whole could have feasibly been compromised.

**Previous Work**

Previous Work in CSfC

This layered approach to implementing security mechanisms as dictated by the CSfC program has some drawbacks. Perhaps most intuitively, there is an increased level of complexity which has some impact on the overhead required to manage the security measure. [3] While the metric of the computational overhead costs of an implementation may, at first glance, seem to lack relevance to the problem of security, a return to the objective of CSfC being in part to offer lower cost solutions for information security needs, makes it of the utmost importance. The computational overhead factors into the operating cost of a solution and thereby may dramatically effect the cost benefit analysis conducted when choosing a security solution. If the overhead is sufficiently costly that it makes a layered solution as expensive to deploy as a GOTS alternative, then there is no reason to assume the extra risk potential presented by the lower level of assurance in commercial solutions. Work has been done to show that this overhead is manageable, with only a marginal percentage increase on security measure deployed individually. [3] This contribution helps to support the notion that layered solutions make sense, at least in that they are cheaper to deploy than GOTS solutions.

Financial cost is not the only factor taken into consideration when selecting a security mechanism, and likewise, computational overhead may not be the only cost of adding complexity as higher levels of complexity have been linked to increased risk in Information Technology security applications. [4] In this use case the addition of complexity, in the form of multiple layers providing a single security trait, also creates difficulties for accurately evaluating the risk presented by a solution. [2] Contributing largely to this difficulty is the interdependence between layers, which is the phenomena that describes a state of shared attributes wherein a vulnerability that effects one layer, may also affect another. [2] Work has been done attempting to create a risk analysis model which could take into account this degree of independence. [5] This work was very broad in its approach and while creating a framework from which the problem could be addressed, failed to provide any concrete measurements of risk.

Further work was also done attempting to link the degree of independence to a measurable change in the amount of risk in a solution. [6] This study began the work of applying quantifiable traits to account for difficulties in the risk analysis process. This is an imperative step in the operationalization of effective cost benefit analysis for layered solutions. While the results of this work were well grounded in simulation and statistics, they lacked a certain degree of tangible, real world context.

Our work will attempt to do for the risk analysis portion of the cost benefit analysis process, what the work on overhead did for the cost evaluation side. That is, by conducting a case study of a layered solution test case, and observing when there is an overlap of critical vulnerabilities, we can observe through empirical historical data, the concepts that were described in theory through the previous INSuRE risk analysis papers. In doing so, we attempt to create a link between the current theories in academia and practical application so that these theories may be verified or revised accordingly.

Previous Work in Vulnerabilities

In order to evaluate the security of the layered solution, we will need to know what vulnerabilities have been published for the mechanisms that comprise it. This may not be the most difficult part of the project as there are national level repositories of this type of information. The most well known repository is the National Vulnerability Database, or NVD. [7] The NVD contains information on vulnerabilities dating back to 2002 across a wide range of products. There are seven details per entry that are relevant to this project. 1) The Common Vulnerabilities and Exposures (CVE) [8] identifier is useful in finding this vulnerability elsewhere, as CVE is a widely used identification scheme. 2) The list of vulnerable products and their versions allows us to filter out only those vulnerabilities that affect the mechanisms relevant to our problem. 3) The date this vulnerability was published shows us the starting point of this vulnerability. 4) A rating by the Common Vulnerability Scoring System (discussed later in this text) [9] gives us a concrete rating for how severe this vulnerability is and what kind of compromises it may have created. 5) A Common Weakness Enumeration (CWE) [10] identifier that provides a scheme to qualitatively categorize these vulnerabilities. 6) References from official sources citing this vulnerability that allow for an official discussion as well as potential patch notes on this vulnerability. 7) A summary detailing the vulnerability and what caused it/what damage it may have caused.

In terms of this problem there are three main systems for determining what a relevant vulnerability is in this context. These systems include the Common Misuse Scoring System (CMSS) [11], the Common Configuration Scoring System (CCSS), and the Common Vulnerability Scoring System (CVSS) [12]. These three are similar; their base metrics all consisting of the access vector, the access vector’s complexity, the levels of authentication, the confidentiality impact, the integrity impact, and the availability impact. Their differences lie in the temporal and environmental aspects. Their temporal metrics are different because the systems have different uses ranging from misuse to vulnerabilities to configurations, while the environment metrics differ due to their providing context for the issue at hand. Because of this those temporal metrics are less important in our scope. The base metrics are the key factors that will be needed. The similarity of those base metrics serves only to indicate that they are one of the better ways to judge the severity of these vulnerabilities. Because those base metrics deal in classifying the severity and outcomes of a vulnerability they will immediately provide value. This problem involves an analysis of vulnerabilities and, due to that, the system that fits our project most closely is the Common Vulnerability Scoring System.

Though it is widely used and it is typically indicated that the CVSS is a good metric for scoring vulnerabilities, [13] it is not perfect. When surveyed, experts indicated some discrepancy in their scoring of the vulnerabilities as a whole; this was found to indicate that the rating of the vulnerabilities need to be contextualized. These experts, in addition to recommending a contextualization of the vulnerabilities, recommended that the CVSS attempt to take into account the specifics of both the environment and the vulnerability instead of classing them into broad categories. In addition, experts found that “Complete impact of Confidentiality, Integrity, and Availability typically means that the vulnerability enables arbitrary code execution” (p. 24). This is one of the major points that our project needs to take into account. Arbitrary code execution immediately makes a vulnerability relevant to our project as it compromises the security of the affected layer. Overall, it solidifies our initial view that leaving this decision of what is useful and relevant to the end user will contextualize the vulnerabilities being analyzed in a much cleaner fashion, making the project as a whole more relevant as well as fixing the initial issue of a lack of contextualization found by the authors.

Our project is built on three main components: a usefulness to the Commercial Solutions for Classified program, a look at vulnerabilities in many environments, and a classification of severity and relevance of those vulnerabilities. Commercial Solutions for Classified allows users of these programs to take advantage of them and their benefits to help reduce the cost of technology. This is where layered solutions begin to come into play. The National Vulnerability Database and the details it provides allow for an analysis of these layered solutions using those details as well as one of several metric systems. One such system, the Common Vulnerability Scoring System, has been shown to be a strong measure of the severity of vulnerabilities as well as a good fit for our project.

Previous Work in Patching behavior

In addition to the publication of vulnerabilities, our work is also concerned with when these vulnerabilities are resolved. If one were to consider the publishing of a vulnerability for a layer the mechanism by which that layer transitions from a state of security to a state of vulnerability, the patching of that vulnerability is the mechanism by which that layer transitions back into a state of security. The time it takes from when a vulnerability is published to when it is patched will be referred to as the notion of "time-to-patch" and has been addressed to some extent in academic writings. In 2004, a research team from Carnegie Mellon University published one of the more thorough works that covers the subject. In attempting to quantify the impact that the disclosure of vulnerabilities has on the speed at which they are patched, they measured the "time-to-patch" of vulnerabilities reported in the CERT/CC database that their university maintains. [15] Their study covered data from 1999 to 2003 and they reported the average time to patch vulnerabilities in those years between 66.19 and 179.84 days with an average over the five year study of 103.36 days. [15] While this study was thorough, it was not primarily interested with describing the "time-to-patch" behavior outside of outside of trying to link underlying causes to it. While the data is still very good, the fact that the base data they used for their measurements is over a dacade old limits its usefulness for out purposes. The same authors published another paper on the subject in 2010. [16] This work was a bit more streamlined and also reported an average time to patch though in this case of 51.4 days. [16] However, this study is still not attempting to describe "time-to-patch' individually, but rather measure again the influence of vulnerability disclosure on this behavior. [16] While this paper still provides good data, it is hindered in that the base date used to generate their measurements was taken in 2001. [16] It is worth noting at this juncture that this paper, while published in 2010, was first submitted in 2006, and approved in 2009. [16] While it is possible if not likely that not all academic papers are separated by so much time between submission and publication, the inherit time lag presents some difficulty in the search for recent data.

The immediate solution to this publication speed concern was to transition to industry papers which report on the behavior. This appeared to make more recent data available at the cost of knowing some of the underlying measurements and processes that went into generating it. One such study was a report from Kenna Security which observed that security teams take over 100 days to close existing vulnerabilities. [17] While the report is from 2015, and in that sense contemporary, they are primarily trying to present the rate at which vulnerabilities are published and the origins of their data are not clear. Additionally, it is unclear whether they are referring to the vulnerability being fixed on the vendor side or consumer side. Another report from Flexera/Secunia claimed that 83.6% of published vulnerabilities were patched on the vendor side on or before the day that they were published and that an additional 1.1% were patched within the first month. [18] While this data is encouraging from a security point of view, no behavior is described for patching beyond the 1 month threshold. Furthermore, the report mentions that the methods used to calculate these measurements, which themselves were not described, had changed making the measurements incompatible with previous reports. [18] This notation, combined with the lack of coverage for vulnerabilities not patched within a month, and the percentages that seem almost too good to be true when compared with the other reports ultimately seem to outweigh the benefit of having data from 2015.

The data that was identified to fit the needs of the project the best was found in a 2015 report by NopSec. While this report still appeared to deal with the notion of "time-to-patch", it provided information at a much finer level of granularity. First, instead of supplying an overarching average time to patch, it broke the industry down into four segments, financial, education, healthcare, and cloud/IT and provided average time to patch for each of these segments individually, observing substantial differences between them. [19] In addition to breaking the average time to patch into these four market segments, NopSec broke down the percentage of vulnerabilities that where patched within specific time periods. [19] Specifically, for the cloud/IT market segment, which encompasses the products that this study is concerned with, 51% of vulnerabilities were patched in the first 10 days, 44% outside the 10 day mark but inside 30 days, and additional 4% were patched before 180 days and 1% from that cutoff to 365 days, with no vulnerabilities taking more than a year to resolve. [19] While this report still is lacking in a certain regard with its explanations of the origins of the data and how it was processed, the insight granted by the distribution of patches allows us to begin to build a model for the patching behavior which can be incorporated into our study.

Ultimately the use of statistical data instead of observed empirical patch dates as intended was unfortunate and changed slightly the intent of the work, but was necessitated by an inability to locate workable sources of information for patch dates pertaining to the chosen products. In spite of this, the location of a high resolution description of patching behavior using recent data, as found in the NopSec report will allow the study continue and still make some reasonable conclusions about the performance of a layered solution.

Conclusion

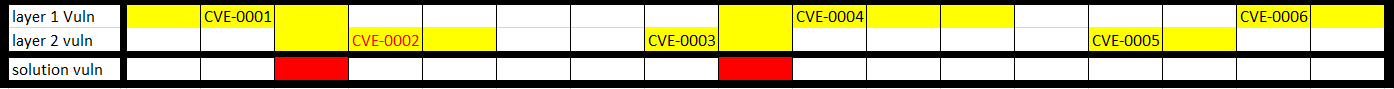
In summary, the body of available research on the Commercial Solutions for Classified program has been evaluated. In spite of making strong contributions to the field, much of this work is largely academic, and while well grounded in theory, is weakly linked to real world examples and empirical data. In the more practical sphere of information technology, we have identified a number of metrics for cataloging the nature and severity of vulnerabilities, as well as sources which provide a comprehensive reference of known and published vulnerabilities. These resources are likely invaluable to operators of information technology systems, enabling them to better protect their charges, but they provide only static information and an initial analysis of the vulnerability in isolation. There is, however, a gap in the literature, in the sense that there is no middle ground between the applied space of the vulnerability databases and scoring mechanisms and the theoretical space of the academic research surrounding the CSfC program. It is the opinion of this research team that the field as a whole would benefit from work in this middle ground. Not only would it serve as a mechanism to ground the theoretical work to empirical data, but it would also allow for more diverse application of the applied datasets. In order that this gap in the research may begin to be filled, and the state of the art in this subject matter be advanced, the following research question is proposed: to gauge the assurance of a test case layered security solution consisting of two redundant mechanisms, is it possible to, using observed vulnerability publish dates and statistically generated patch dates, simulate the performance of the solution and infer the relative reduction it provides in terms of system vulnerability.?

**Expanded Problem Statement**

The high level objective of our project is to observe and describe windows of vulnerability in the lifecycle of a layered solution. We define a window of vulnerability to be a contiguous string of one or more days when a layered solution is observed to be in a state of vulnerability. In order to do this we will produce a timeline spanning the entire lifecycle of this solution, recording at each day in that timespan whether or not the solution is in a state of vulnerability. For our purposes, we will consider the lifecycle to be a span of two years (pending TD approval).

The layered solution to be observed will be comprised of two secure VPN products. These products will be selected based on four criteria. First the products must generally be considered to be secure. Second, these products must be from two different manufacturers. Third, these products must be mainstream, in other words, they should be implemented with a degree of proliferation that their compromise would be noteworthy. Finally, these products should have resources that provide rich information regarding security issues and patch data so that the assessment may have the highest quality of data available when making observations.

In order to build our construct of a window of vulnerability we need to be able to determine through observation if the layered solution is in a state of vulnerability. While this is a key element of the project, this phenomena is not directly observable, but rather a function of the state of the two mechanisms that comprise the solution. For this study, we consider the layered solution to be in a state of vulnerability when all of the mechanisms within the layered solution are in a state of vulnerability. Figure 1 below shows how this might look on a timeline. The yellow blocks in the first two rows show periods of time when each of the two mechanisms (divided by row) are in a state of vulnerability. When these periods overlap, both mechanisms are in a state of vulnerability and the solution itself is determined to be in a state of vulnerability, as indicated by the red boxes in the third row.



*Figure 1*

For this study, we consider a mechanism to be in a state of vulnerability when, at some point in time, there exists an unpatched vulnerability in the mechanism which constitutes a direct threat to the security attribute of the information technology asset that it was intended to provide. For instance, a vulnerability that causes a temporary inability to access information in a mechanism intended to prevent unauthorized access to that information would not necessarily cause a state of vulnerability in the mechanism whereas one that temporarily allowed universal access would. The state of vulnerability in a mechanism caused by a critical vulnerability may also be extended beyond the patch date, or before the publication of the vulnerability in special cases where the vulnerability allows attacker persistence in the solution, or an attack on intercepted data stored before the vulnerability was published. Figure 2 below shows how this observation may look on a timeline. The colored blocks in the first three rows depict times when there is an open vulnerability in a mechanism. An open vulnerability in the scope of this project is considered to be one that has been published but not yet patched. The inclusion of more than one row to represent this data was to accommodate the occurrence of multiple concurrent vulnerabilities. The grey blocks represent open vulnerabilities which were determined to be non-critical, where-as the yellow blocks indicate those vulnerabilities which represent a genuine threat to the integrity and function of the mechanism. Whenever there is an open and critical vulnerability, as indicated in yellow, the mechanism will be determined to be in a state of vulnerability, as indicated by the red boxes in the fourth row.



*Figure 2*

In order to determine what the open vulnerabilities are within a mechanism, we must first find out what vulnerabilities have been published that pertain to it. This task will be approached from two directions, both of which leverage the National Vulnerability Database (NVD). The NVD maintains a comprehensive list of published vulnerabilities along with context information. This information includes what piece of software they were found in. This database can be downloaded and searched and in this way we can find vulnerabilities that occur directly in the mechanism that we are dealing with. This is the first direction in which the publish date of vulnerabilities will be addressed. However, not necessarily all of the vulnerabilities that pertain to a mechanism will exist within the code of that mechanism. Vulnerabilities in things such as libraries and protocols will be linked to that library or protocol and it may not be obvious which ones pertain to the mechanism of interest. Fortunately, companies like Cisco, often publish security advisories for their products when a vulnerability is released which impacts them. This characterizes the second direction from which we will approach the collection of vulnerability publish dates. We can use references from the manufacturer to relevant vulnerabilities to return to the NVD data set and collect information for those vulnerabilities which impact the mechanism, but are not internal to the mechanism itself.

In addition to the publication date of the vulnerability, in order to identify when the vulnerability was open, we need to identify the date at which the patch was released. Such information is not resident in the NVD dataset and its location and content is likely to be specific to the vendor of mechanism. Efforts to identify a source of information for this data were unsuccessful and necessitated a shift in focus from conducting a case study based purely on empirical data to a simulation using both observed and statistically generated patch times to create a possible case of performance for the solution.

With the open vulnerabilities identified, the task was to identify which of them represent a critical risk to the integrity and function of each mechanism. While the specific criteria on which this determination is to be made are still in development, the intent is to leverage information such as the CVSS score and CWE classifier in conjunction with the nature of the mechanism to make these determinations on a case by case basis. After this case to case analysis, a set of search strings was developed to filter out vulnerabilities to be determined not to constitute a break of the mechanism.

Once the timeline had been constructed, basic measurements were taken with regards to the windows of vulnerability for each layer and the solution as a whole. These included the number of windows of vulnerability and the minimum, maximum and average duration of these windows of vulnerability. These measurements, taken after a single observation using a statistically generated patch date for each observed vulnerability, represent one possibility for how the solution may have performed based on the statistical model of patching behavior. In order to better approximate the potential performance of the solution, a large number of these possible cases were generated and their performance measured. The trends of these measurements infer how well the solution could have been expected to perform.

**Significance**

This study will provide a clear way to gauge the assurance improvements of a layered solution by not only identifying the amount of time that each individual security product spent in a state of vulnerability, but also the amount of time the solution as a whole was in a state of vulnerability. While the scope of this case study is limited to two vpn products and does not by any means cover the entirety of the problem space, it is an essential first step which should provide some initial indications as well as serve as a field test of the methods used to collect and analyze the relevant data. Should the data collection and analysis approach in this study prove efficient, these methods could be applied to conduct analysis of other security products and layered solutions without the need to redevelop the mechanisms.

Perhaps as importantly, this work will begin the process of building a bridge between the more academic research already conducted on the topic of layered solutions and the applied data repositories being used as sources for this investigation. This will provide a level of grounding for the academic research and perhaps even provide metrics by which the claims made can be supported. Conversely it may provide a mechanism through which more abstract analysis of the empirical data can be supported.

This project will also help check costs of commercial solutions (CSfC) against that of government developed solutions. The ability to easily conduct such a cost benefit analysis and from that analysis gain important insight into the increase, or lack thereof, of layered commercial solutions compared to government solutions is one that can save a lot of time and effort. If the security of several commercial solutions equals or betters that of a government solution while also being cheaper to purchase and install that layered commercial solution might be a better choice for securing information. Were this study not attempted that time and effort would be invested elsewhere, and these comparisons of commercial vs government solutions would not be nearly as complete or definitive as they are now, and extra money might have been spent on products that were more expensive and potentially not as secure.

**Limitations Delimitations and Assumptions**

General Delimitations

For this study, we assumed that an attacker will only use published vulnerabilities to attack a system. We also assumed that for all vulnerabilities which can be reasonably exploited to break a mechanism, that the attacker has the resources and motivation to accomplish this. We made the further assumption that an attacker does not have local or physical access to the machine, nor do they have the assistance of anyone with local or physical access to the machine, whether knowing or unknowing. This precludes the inclusion of any vulnerabilities which are characterized as needing local access to be exploitable, or being exploitable over the network with user assistance. We assume for the case of the static or persistent exploits that the attacker stores all data that they can get, so that they may attack it in the future if the opportunity presents itself. We assumed that the defender is doing nothing to protect their system outside of patching. Furthermore, we assumed that the patch for a particular vulnerability will be applied one week after the patch has been released. While the defender is doing nothing to defend their system aside from patching, it is also assumed that they are following best practices for maintaining a secure system and are not being negligent in managing their systems or otherwise undermining the system in a way which may be considered as providing user assistance by the vulnerability catalogues.

For our analysis, we also assuming that the descriptions of the vulnerabilities to be found in the national databases are accurate and complete. We also assume, barring future vulnerability releases, that the patches applied to fix vulnerabilities work, and will consider the vulnerability closed after they have been applied. Additionally, this study making the implicit assumption that the two mechanisms selected to represent the layered solution are representative of all such mechanisms. This is perhaps the weakest assumption of the study and will likely require follow on work to confirm or modify any findings, but it is necessary given the time and resource constraints placed on this work.

In terms of the statistical model for generating patch dates we are limiting the distribution to basing it off of the Cloud and IT section of information from the NopSec report. [19] This data was used to develop the distribution model used for generating patch dates. In using this data in such a way, we are making the implicit assumption that the data provided in the model is accurate and representative of industry trends.

Ongoing Limitations

Some difficulties were encountered throughout the course of the work which limit the effectiveness of the study. The biggest of these is the complication of finding accurate patch or fix dates for the vulnerabilities identified. While there is abundant information regarding the publish dates of the vulnerabilities and the nature of the threat that the represent, it is difficult to determine when a fix was first made available. As finding empirical patch dates was not a viable avenue, a statistical model for generating patch dates was created. This model is based off of the paper by NopSec and is limited by the accuracy of that paper. [19]

Additionally, there was some difficulty with regards to vulnerabilities which may exist within the software dependencies of particular security mechanisms. The current search strategy which is being implemented only identifies vulnerabilities which exist within the code of the security products themselves, not in any underlying libraries or protocols.

Assumptions

There are several assumptions that were made out of necessity for this project, the majority of them stemming from the statistical analysis of patch dates. The first is that each of our layers are interchangeable within both the script and the gamma distribution. This simply means that the gamma distribution we chose to use to generate patch dates for our vulnerability works equally well with every layer. This assumption was made so that the project could move ahead with the use of this distribution to generate the patch dates. The second assumption is that the aforementioned gamma distribution is the correct distribution to use. After consultation with the Purdue University Statistics Consulting Office this distribution was determined to be best for this project. While the consulting service oversaw the fitting of this distribution, there is the underling assumption that the data from the NopSec report that it was fit to is correct and representative of the trends in industry which it describes. [19] Additionally, the assumption has been made that the two vpn products, which are the subject of the study are represented by the cloud/IT measurements in the aforementioned study. Finally, we are assuming that all of the vulnerabilities observed are independent events and follow the patching behavior modeled uniformly, regardless of contextual factors. This again may be a rather weak assumption but was necessary to apply the statistical data in the manor that this study did.

**Methodology**

Identification of Layers

The general framework applied to this work was that of an observational case study. The subjects of this case study were to be two security mechanisms, specifically, secure VPN applications. These mechanisms were to represent the two layers of a layered solution mock up. The mechanisms chosen for this case study were Cisco AnyConnect and Juniper Pulse, which is now maintained by PulseSecure llc. [20][21][22] These products were selected via a non-probability sampling scheme, specifically, judgment sampling. [23] Under this scheme they were identified as fulfilling the requirements, outlined in detail in the expanded problem statement, of being marketed as secure, from different manufacturers, having a large market share and having rich information available. While such a selection scheme limits the statistical significance of the results, choosing well known products will improve the plausibility and relatability of the case study and choosing products with as much information available as possible not only makes the development of data collection and analysis strategies easier, but helps to ensure that the data used is accurate, reducing the inherit level of variance in the results.

Identification of Vulnerabilities within Layers

After choosing layers we identified all vulnerabilities within those layers. As discussed earlier, our data source of choice is the National Vulnerability Database (NVD) [7]. The seven details (The Common Vulnerabilities and Exposures (CVE) [8] identifier, the list of vulnerable products, the date this vulnerability was published, a rating by the Common Vulnerability Scoring System (CVSS) [9], the Common Weakness Enumeration (CWE) [9] identifier, References from official sources citing this vulnerability, and the vulnerability summary) allow us to easily associate each vulnerability with its respective layer. This NVD vulnerability information is provided as an XML file. At this moment in time there are approximately 77,000 vulnerabilities within the NVD's data set to analyze and parse for relevancy. We parse through the data for each vulnerability pulling those seven main data points. Utilizing the list of products each vulnerability affects we can gather only those vulnerabilities that affect the specified layers. The arguments pased into the search functionality of the program for the primary search where "Cisco AnyConnect" and "Juniper Pulse". This allowed the search function to identify products starting with the string "AnyConnect" from the manufacturer Cisco and products starting with "Pulse" from the manufacturer Juniper. Further analysis showed that the Pulse vpn functionality from Juniper was listed under several different product names within the NVD including Junos Pulse in 2014 and it was tagged as part of the IVE network OS prior to that. In 2016 it was spun off and identified as being a product of PulseSecure. An alternative name input was added to the script to allow for it to handle cases of this type.

Filtering of Identified Vulnerabilities

In addition to gathering the vulnerability information, the NVD description for each vulnerability was parsed for relevance to the condition defined at the beginning of the project and later confirmed by the technical director. This condition is that the vulnerability constitute a complete break in the security provided by the mechanism. The Common Vulnerability Scoring System (CVSS) information contained within the vulnerability entry was leveraged for its built in metrics as well as the short summary of the vulnerability included in the NVD entry. [14] The CVSS information includes scoring on the access vector which was used to eliminate vulnerabilities which required local access to the target machine, a case ruled out of scope for this work, by ensuring the this value within the vulnerability description was either a 1 or 2, and not 0. In addition to filtering out the vulnerabilities which require local access, a keyword search was added to scan through the description of the vulnerability to look for functionality which would constitute a break of the security mechanism. The determination was made that such vulnerabilities should include privilege escalation, credential stealing, session spoofing, and arbitrary reads and writes controlled by the attacker. In order to filter so that only these vulnerabilities were assessed, each vulnerability still relevant after the first level of filtering was analyzed by hand in order to determine which could fulfill the above criteria. When all of the vulnerabilities with such functionality were identified, they were used to create search strings which could be identified by the filter function so that all in this limited case would be caught. The strings chosen were "bypass intended access restrictions", "valid meeting ids", "arbitrary", "clickjacking", "gain privileges, root, capture credentials". While these strings were chosen to select a predetermined subset of database entries with perfect accuracy for out test cast, the functionality in the script allows for more generalized cases to be run in the future.

After filtering for relevancy, three vulnerabilities were left in Cisco AnyConnect and five in Juniper Pulse.

Application of Patch Dates

Much effort was applied to the objective of obtaining patch dates for the vulnerabilities that were identified, but that part of the study was, in the end, unsuccessful. The decision was made that instead of using observed patch dates, statistical methods could be applied to approximate this behavior. Part of this process was identifying a method to randomly generate patch dates that follow the patching behavior of the industry. The data chosen to build this functionality off of was the NopSec 2015 report. [19] This choice was made do to the increased level of resolution that this study provided in terms of its rough histogram of probabilities. [19] The team met with Yixuan of the Purdue University Stats Consulting office with this data and a description of the problem in hand. At his direction a gamma distribution was fit to the data using the maximum likelihood methodology and the following equation.



*Figure 3*

This equation was optimized using the Nelder-Mead reduction method and the gamma library of SciPy, under the supervision of Yixuan to yield values alpha = 0.875424 and theta = 15.592681 as the best fit for a gamma distribution.

Using this distribution, random patch dates which follow the distribution of the time to patch behavior for the cloud/IT market segment as reported by the NopSec paper. This can be accomplished in the SciPy extension of the Python scripting language by using the gamma.rvs() function, using the alpha and theta parameters as input. This function call took the place of the method that was being reserved in the implementation for retrieving the patch dates. This can be replaced without hassle in the future, but as it the script was used for this study, when it attempted to retrieve a patch date for a vulnerability, it instead generated a random time to patch in days using this function.

Simulation

In order to infer the performance of the proposed layered solution, a simulation was run to generate a large number of cases from which measurements could be taken and analyzed. The observation period for which the performance of the solution was to be evaluated was chosen to be two years long and last from 1 January, 2014 to 31 December, 2015. The length of two years and the start and end dates were selected in part to make the results easier to interpret and reduce the number of alternate identifiers which had to be supplied for the pulse product.

Each case involved plotting the vulnerabilities for each layer which were published during the two year observation period and fulfilled all the requirements set forth for the study. Other vulnerabilities were omitted through the filtering process described above. For each one of these vulnerabilities a time to patch offset was generated using the statistical method described above. The close date for the window of opportunity created by the vulnerability was set at this offset from the publish date, plus seven days as per the time to implement parameter set in the project description.

The windows of opportunity created by each vulnerability within a layer were condensed so that overlapping windows created by multiple vulnerabilities in the same layer become a single window using the earliest publish date of the overlapping group as the start time and the last patch offset as the end time.

For each layer, three measurements are taken, the number of days over the course of two years that the layer was vulnerable, the number of windows that were present during that two year period, and the longest window of opportunity that was observed.

Windows of opportunity for the solution as a whole were plotted for every span of time of time were AnyConnect AND Pulse had a window of opportunity. The same three measurements of number of days vulnerable, number of windows of opportunity, and longest window were taken.

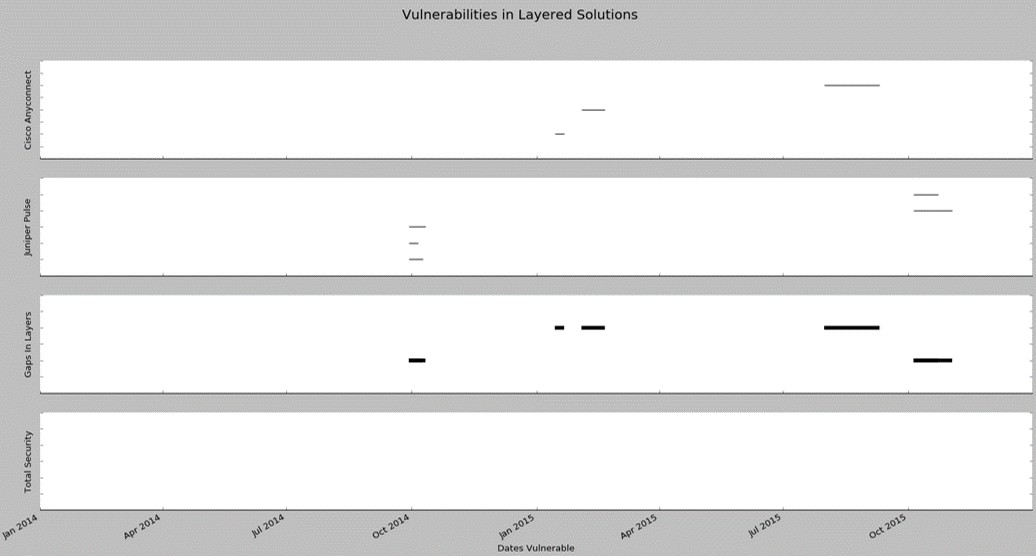
While each test case and the nine measurements collected represent one possible outcome of how the system may have performed, in order to get a more accurate assessment, 100,000 cases were run and the measurements from each were recorded. The number of cases was suggested by the Stats Consulting Service.

The recorded measurments for all 100,000 cases were stored and subjected to basic descriptive statistics in order to describe the expected performance of the simulated layered solution.

**Implementation**

Our implementation of the aforementioned methodology is done in the Python language, more specifically Anaconda. Anaconda is used for easy setup of the various libraries needed including MatPlotLib and SciPy. The script has six main components.

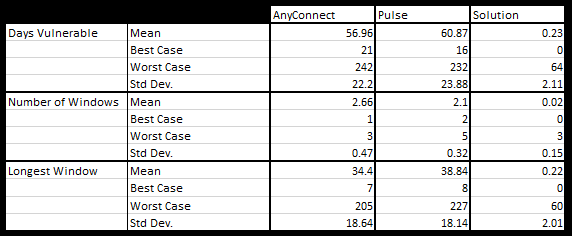
1. **Read the command line arguments**. There are several parameters that can be passed in to the script to help it gather the vulnerabilities that are relevant for that run. A few will be discussed here, but they are also detailed in comments in the script itself.
2. **Gather vulnerabilities from the National Vulnerability Database**. Using the layers specified the script can begin to pull vulnerabilities from the NVD files. In addition to the names of the layers specified, the script can take as input alternative names for each layer. For example, Juniper Pulse has gone by several different names including “Juniper Pulse Connect Secure” and “Juniper Junos Pulse Secure Access Service”. By allowing these alternative names we are able to gather a more complete set of vulnerabilities for our layers, especially if these layers have changed in name. In addition to this the script allows for the specification of a start and end date. These dates indicate the period of time that the script should gather vulnerabilities from. If the user is interested in vulnerabilities from 2014 they would specify the corresponding start and end dates and the script would gather only vulnerabilities from 2014. This NVD vulnerability information is provided as an XML file within a zip archive. By utilizing Python's request, zipfile, and io libraries we can automatically download, unzip, and begin analyzing those XML files
3. **Apply the filters specified on the command line to the gathered vulnerabilities**. At the start of the script, the user is able to input the minimum level for each of the Common Vulnerability Scoring System metrics. Vulnerabilities that do not meet those minimum specified metrics are not considered relevant within that run of the script as they would not constitute a breach of that layer. In addition to specifying the minimum CVSS metrics the user can also specify a set of keywords that they would like used as additional filtering. A vulnerability's summary must have at least one of the specified keywords in its summary in order to be considered relevant. If none of the keywords are found in the summary the vulnerability is not considered relevant within that run of the script. After this filtering takes places we have a list of all vulnerabilities that, as specified by the user of the script, are considered relevant.
4. **Estimate a patch date for each vulnerability remaining**. The script utilizes the SciPy library and a random number generator based off of a gamma distribution to estimate a patch date for each vulnerability. This was implemented into the script in such a way that, if empirical data or more accurate statistical data is found, it would be extremely easy to change out the patch dates on each vulnerability.
5. **Obtain measurement data on the windows of opportunity through the use of the gamma function**. The patching simulation is run a large number of times (100,000 for our tests) to obtain measurements as to the windows of opportunity. For each layer and the system as a whole information is gathered as to the total days vulnerable, the number of windows of opportunity, and the longest window of opportunity. This allows us to make inferences as to when each layer and the system as a whole would have been vulnerable within the parameters specified in the command line.
6. **Plot the relevant vulnerabilities**. Using the MatPlotLib library we can plot a single run of the script at a time. This visualization includes plots representing vulnerabilities within each individual layer, windows of vulnerability for each layer, and windows of vulnerability for the system as a whole. This allows for an easy way to show when the layers and the system were or were not vulnerable. For example, after gathering all vulnerabilities for both Cisco AnyConnect and Juniper Pulse within the time period of January 1st, 2014 to December 31st, 2015 the results show 3 relevant vulnerabilities for Cisco AnyConnect and 5 relevant vulnerabilities for Juniper Pulse. These are laid out in the plots below, with the top two plots being the individual vulnerabilities affecting Cisco AnyConnect and Juniper Pulse. The second to last plot represents the windows of opportunity for each of the two layers identifying where in time those two layers individually are vulnerable. For Cisco Anyconnect there are three windows of opportunity in this two year period. For Juniper Pulse there are two windows of opportunity in this two year period. The bottom plot is a representation of the windows of opportunity for the system as a whole. Within this simulation this plot is empty, signifying that the system as a whole has no windows of opportunity.



*Figure 4*

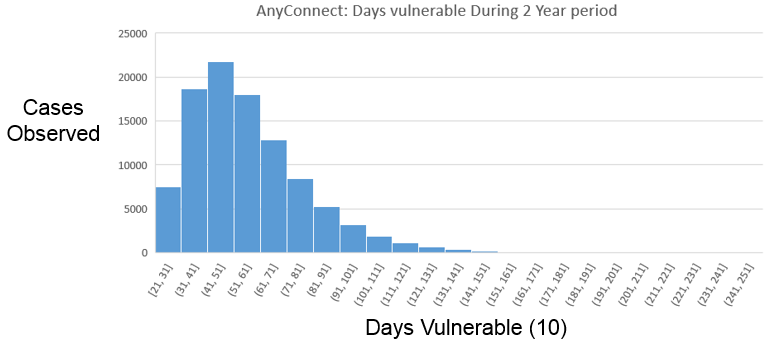
**Results**

The 900,000 measurements collected from the simulation cases were analyzed using basic descriptive statistics. The results of this analysis are included in figure 5.

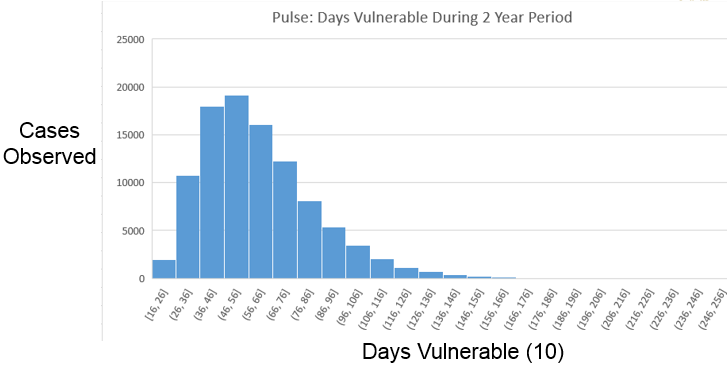


*Figure 5*

The number of days that each layer and the solution as a whole was chosen as the general descriptor of performance for the purpose of presentation and this data was charted in a histogram with the number of cases observed on the Y axis and the number of days vulnerable on the X. The data for AnyConnect is shown in figure 6 and Pulse in figure 7.



*Figure 6*



*Figure 7*

The histogram for the solution for the days vulnerable data was omitted. 98.16% of the cases run or 98,160 cases had no overlap in vulnerability between the layers resulting in the solution not being vulnerable during the observation period. While interesting data, this makes for a very hard to read histogram.

The simulation clearly showed that the solution spent less time vulnerable than either one of the layers individually. In terms of ratios, the solution itself, on average spent 99.6% less time in a state of vulnerability than AnyConnect and 99.63% less time than Pulse. So at least in this case, using these simulation parameters, the benefits of opting for a layered solution were substantial. As mentioned above, in over 98% of the cases that were observed, the solution spent no time at all in a state of vulnerability, negating attacks using only published vulnerabilities completely.

**Conclusions**

In the narrow sense, the experiment that was conducted in this work showed that operating two security mechanisms as a layered solution, specifically the VPN products Cisco AnyConnect and Juniper Pulse, over the time period of 1 January 2014 to 31 December 2015 would have likely recused the time that the system spent vulnerable substantially, if not negated the vulnerability surface completely for an attacker who only uses published vulnerabilities for the products themselves, forgoing 0 day vulnerabilities and exploitation of other elements of the system. This is assuming that the way we chose to model that patch times was accurate and representative of the behavior in these products and that the vulnerabilities we identified as being a critical threat to security were the only ones that represented such a threat.

However, this outcome was not necessarily a surprise, intuitively, it makes a lot of sense that a layering approach would reduce the treat windows, the real question is by what degree. While this single case looked very promising, it does not and really cannot say a great deal about the performance of layered solutions in general, and that was not truly the point of the exercise.

Perhaps the greater contribution of this work is the development of the scripts to harvest and analyze the data to make these assessments. Through this work we have built and field tested a largely automated methodology by which assessments regarding the performance of layered solutions can be assessed. This provides the groundwork for future teams researching the subject to work with a higher level of effieciency and takle some of the more interesting problems.

**Discussion**

The research question that this project set out to answer was, "to gauge the assurance of a test case layered security solution consisting of two redundant mechanisms, is it possible to, using observed vulnerability publish dates and statistically generated patch dates, simulate the performance of the solution and infer the relative reduction it provides in terms of system vulnerability?" Though there were a few issues in answering this question the answer is a definitive yes.

Though there was difficulty acquiring empirical data regarding patch dates the use of statistics made up for that lack of data and provided the best alternative. Additionally, the criteria that are being used to filter out those vulnerabilities which are not relevant from the ones that are may not be ideal, or applicable in every scenario in which one might wish to gauge the effectiveness of a solution. However, every effort has been made so that the filters can be adjusted, through the course of this project and for anyone who whishes to reuse the implementation so that the filters can be optimized for the given use case. The fact that no vulnerabilities were identified for the Pulse VPN package before 2015 is also a matter of concern, though we are assuming that our NVD dataset is complete, so it is the data we have to work with.

Looking at the graphed results, it is clear the layered solution is in a state of vulnerability less often than either of the individual layers. While this is the expected outcome and almost certainly the case outside of all of the windows of opportunity having perfect alignment in both layers, it is a state that could have been intuited by looking at the nature of the problem, and not very informative. The real value lies in determining exactly what the benefit is of the layering, how much less time is spent in a state of vulnerability, which at the moment is impossible to determine as our fix dates for the vulnerability are set arbitrarily with mock data for the sake of presentation and function testing the timeline. The theoretical mean of the windows of opportunity from publish to patch generated by the gamma function is within 10 days as per the NOPSec report. [19] These statistically generated patch dates allow us as close to a real world approximation as is allowed in the absence of real empirical data.

Even assuming that the data collection and analysis works flawlessly, this study is still limited in that it is observing only two pieces of software, both of which are VPN solutions, and chosen based upon the rather stringent criteria outlined twice in this report. As such, the observations made in this study may not generalize well to other types of software, or software that does not meet the selection criteria of this study.

The real value of this study is the methods, scripts, and statistical model developed to tackle this problem. Unlike the specific results, these mechanisms are agnostic to the type of software, and selection criteria specific to this project. In addition to making novel use of existing information technology data sources, careful selection of products for each layer can be used to test the claims of risk interdependence proposed in previous insure projects with real world data. [5][6] In so doing, this work will have succeeded in attempting to bridge the more theoretical academic works with practical application.

**Future Work**

There are two main avenues for future work on this project and several smaller avenues. The first main avenue is to actually obtain empirical patch dates. With that information this study could be repeated for a large number of potential solutions in order to observe the historical behavior of these solutions to make empirically grounded statistical inferences about the behavior of such solutions in the future. The second main avenue of future work for this project would be to replace the empirical publish dates with statistically generated ones. By converting the entire project over to a statistical model one could simulate or formalize a generalized approach to estimating the benefits of layered solutions. Essentially convert this study to entirely use empirical data, or to entirely use statistical data. This will even out the results a bit due to the uniformity of how the data is generated. It will also allow for several different conclusions to be drawn that can not be drawn through the current methods used.

One smaller avenue of work surrounds finding information regarding the availability of exploit code. ExploitDB is a website that houses vulnerability exploit code for many known vulnerabilities. These exploit code entries are initially unverified, though as they are tested they gain a verified status declaring that they really do exploit the vulnerability they claim to. Unfortunately Exploit DB has a captcha on their website, so automating this part is not possible. If the number of vulnerabilities that our script returns is small this would be trivial to manually look up.

In addition to the above, refining the parameters for this simulation to more realistically reflect the layers being investigated would drastically increase the accuracy of the results generated. If there are additional filters that we did not implement adding those would also further increase the accuracy of the results generated.

In terms of the code from the script itself, some of it is less optimal that it could be due to the lack of time we ran into at the end of the semester. It can be optimized so that it will run faster.

**Deliverables**

Upon completion of this project or the conclusion of this semester we are tasked with providing:

* Project Proposal
* Final Report analyzing Outcomes and Methodologies
* Poster Outlining Project, Methodology and Findings
* Presentation of Project Outcomes and Methodologies to the Sponsoring Parties
* Timeline representing “holes” in each individual layer
* Scripts used for data collection and analysis
* List of vulnerabilities for each layer along with the relevance of each vulnerability

**Issues**

There are several issues that our team faced during the duration of this project. One of these was the lack of information surrounding relevant vulnerabilities for the solutions chosen. Building upon this, the main piece of information that was missing were the concrete patch dates. NVD does have entries in its files that specify when each vulnerability was patched, and although there are companies that do have that information from our experience they are all behind some form of corporate paywall. It is the same with the companies who's products we are looking for vulnerabilities in. Typically unless one has a service contract with said company the patch data is unavailable. In addition, this lack of empirical data caused us to spend a large amount of time talking to various entities in an attempt to find that empirical data. This unfortunately cut down on the time we had left in the rest of the semester and although it was not a huge issue we did not make much progress in our search for those empirical patch dates.

There was a slight issue in working with SciPy, though that was fixed by using the Anaconda version of python which comes with all of the key libraries our script needs already installed.

**Bibliography**

[1] “Commercial Solutions for Classified Program” National Security Agency, Information Assurance. 2012 <https://www.nsa.gov/ia/programs/csfc_program/>

[2] F. Roeper, N. Ziring “Building Robust Security Solutions Using Layering and Independence” RSA Conference 2012. 2012 <http://www.rsaconference.com/writable/presentations/file_upload/star-401.pdf>

[3] C. Farrington, D. Hlavacek, E. Rodine, C. Spear. “Commercial Solutions for Classified Layering Costs” Iowa State University as part of the INSuRE Project. 2014. Available at https://purr.purdue.edu/projects/insureclass/files/browse?subdir=Projects/Commercial%20Solutions%20for%20Classified%20Layering%20Costs/Fal2014%20ISU%20Farrington-Hlavacek-Rodine-Spear/Final%20Report

[4] “Examining the Dangers of Complexity in Network Security Environments” AlgoSec Survey Insights. 2012. Available at http://www.algosec.com/resources/files/Specials/Survey%20files/12\_10\_11\_security\_complexity.pdf

[5] C. Martinez, R. Haverkos. “CSfC Risk Analysis” Purdue University as part of the INSuRE Project. 2014. Available at <https://purr.purdue.edu/projects/course14fall/files/browse?subdir=Projects/Purdue/Risk%20Analysis%20CSfC/Final%20Report>

[6] J. Zage, R. Wells, M. Farnam "Risk Analysis of a Layered Solution", Purdue University as part of the INSuRE Project. May 1 2015. Available at https://purr.purdue.edu/projects/insureclass/files/browse?subdir=Projects/Risk%20Analysis %20CSfC/2015Spr%20Pur%20Farnam-Wells-Zage

[7] “National Vulnerability Database”. NIST. 2002. Available <http://nvd.nist.gov>

[8] “Common Vulnerabilities and Exposures”. MITRE. 2014. Available <https://cve.mitre.org/>

[9] “Common Weakness Enumeration”. MITRE. 2015. Available <https://cwe.mitre.org/>

[10] K. Scarfone, P. Mell, S. Romanosky "A Complete Guide to the Common Vulnerability Scoring System Version 2.0" June, 2007. Retrieved from <https://www.first.org/cvss/cvss-v2-guide.pdf>

[11] J. R. Crandall, D. Oliveira. “Holographic Vulnerability Studies: Vulnerabilities as Features in Interpretation as Information Flows Across Abstraction Boundaries” NSPW ‘12. 2012. Available at <https://www.cs.unm.edu/~crandall/nspw2012.pdf>

[12] S. H. Houmb, V. N.L. Franqiera, E. A. Engum, “Quantifying security risk level from CVSS estimates of frequency and impact,” *The Journal of Systems and Software, 83*, pp. 1622-1634. August 2009. Retrieved from <http://ac.els-cdn.com.ezproxy.lib.purdue.edu/S0164121209002155/1-s2.0-S0164121209002155-main.pdf?_tid=6553d620-d27d-11e5-9919-00000aab0f27&acdnat=1455387351_61e169e87aa54f8602e28be7b84a7e4a>

[13] H. Holm, K. K. Afridi, “An expert-based investigation of the Common Vulnerability Scoring System,” *Computers & Security, 53*, pp. 18-30, May 2015. Retrieved from

<http://ac.els-cdn.com.ezproxy.lib.purdue.edu/S0167404815000620/1-s2.0-S0167404815000620-main.pdf?_tid=d1c6c39a-d27c-11e5-b44a-00000aacb362&acdnat=1455387104_c1af4dd79ed1130cd821c3b310b74391>

[14] E. LeMay, K. Scarfone, & P. Mell “The Common Misuse Scoring System (CMSS): Metrics for Software Feature Misuse Vulnerabilities,” July, 2012. Retrieved from <http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7864.pdf>

[15] A. Arora, R. Krishnan, A. Handkumar, R. Telang, Y. Yang, "Impact of VUlnerability Disclosure and Patch Avalability – An Empirical Analysis" H. John Heinz III School of Public Policy and Management, Canregie Mellon University, April 2004. Available <http://www.dtc.umn.edu/weis2004/telang.pdf>

[16] A. Arora, R. Krishnan, R. Telang, Y. Yang, "An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure". Information Systems Research, Vol. 21, No. 1, March 2010. Avaialble at <http://web.b.ebscohost.com.ezproxy.lib.purdue.edu/ehost/detail/detail?sid=7717ba0d-6599-4fea-8a2c-8faf2e862100%40sessionmgr107&vid=0&hid=128&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=51884384&db=buh>

[17] "How the Rise in Non-Targeted Attacks Has Widened the Remediation Gap" Kenna Security, September 2015. Available <https://www.kennasecurity.com/resources/non-targeted-attacks-report/>

[18] "Vulnerability Review 2016", Flexera Software, March 2016. Available <http://resources.flexerasoftware.com/web/pdf/Research-SVM-Vulnerability-Review-2016.pdf>

[19] "2015 State of Vulnerability Risk Management", NopSec, 2015. Available <http://info.nopsec.com/rs/736-UGK-525/images/NopSec_StateofVulnRisk_WhitePaper_2015.pdf>

[20] "Cisco AnyConnect Secure Mobility Client". Cisco. 2016. Available <http://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html>

[21] "Junos Pulse Products have Moved to a New Home" Juniper Networks. 2015 Available <http://www.juniper.net/us/en/pulsesecure/>

[22] "Pulse Connect Secure" PulseSecure. 2016. Available <https://www.pulsesecure.net/connect-secure/overview/>

[23] "Non-probability sampling: convenience, judgement and snowball sampling" University of the West of England, Research Observatory. 2007. Available <http://ro.uwe.ac.uk/RenderPages/RenderLearningObject.aspx?Context=7&Area=1&Room=3&Constellation=28&LearningObject=164>

**About the Investigators**

Robert Haverkos is a second year Master's student in the Center of Excellence in Research for Information Assurance and Security (CERIAS) information security program at Purdue University. He has research experience in the fields of steganography and risk analysis with additional interest in the area of malware analysis. He previously worked on the Fall 2013 CSfC Risk Analysis project under INSuRE.

Daniel Sokoler is a junior attending Purdue University pursuing a Bachelors of Science in Computer Science with focuses in Computer Security and Software Engineering. In addition he is in the process of completing the Certificate in Entrepreneurship and Innovation.

**Schedule**

* 17 February - Proposal Submission
* 21 February - Complete Initial Literature Survey
* 24 February - Initial TD Meeting & Project Onboarding
* 11 March - Data collection and analysis plan complete / Mechanisms for test solution selected
* 18 March - Data collection mechanism finalized
* 25 March - Data Collection Final
* 8 April - Analysis Final
* 18 April - Final Report Submission
* 4 May - Project Presentation

**Budget**

Researcher Salary: 360hrs - $7,200

Benefits (22% salary): $1,584

University Overhead (54.5% salary + benefits): $4,788

Conference Submission/Travel: $3,800

Total Budget: $17,372

**Appendix A: Research Conference**

New Security Paradigms Workshop (NSPW, 2016)

26-29 September 2016

Colorado

Submission Deadline 29 April 2016

<http://www.nspw.org/2016/cfp>

Related Paper:

Holographic Vulnerability Studies: Vulnerabilities as Fractures in Interpretation as Information Flows Across Abstraction Boundaries.

J. R. Crandall, D, Oliveira

NSPW ‘12

Overview

The new security paradigms workshop is a rather small conference that takes place in a different location every year, the 2012 offering from which the example paper was selected was held in Bertinoro, Italy, where as the 2016 workshop will be held in C Lazy U Ranch, Colorado. It is run under the purview of the ACM and seems quite small compared to some of the more well known events. The focus is on new or emerging ideas in security and every submission / presentation is intended to be accompanied by a discussion among the participants. This type of setting was determined to be an ideal place to present the work of this project. While the CSfC program and its interpretation of layered security solutions may be a few years old, it seems limited in its exposure outside of its immediate customer base. Such a setting would not only provide an opportunity to field the test case layered solution visualization, but to get industry feedback on the idea as a whole, and its general applicability outside of the government contractor market. The exemplar paper selected by Crandall and Oliveira is not as directly related to the work as Roeper and Ziring presented at the RSA conference in 2012, but the RSA submission dates would mean that the paper would not be eligible for submission until the 2017 event. [1] That being said, the Crandall and Oliveira paper is not without its relevance, at least on the conceptual level. In their paper, they identified a problem with existing vulnerability taxonomies in that they are often limited to a single level of abstraction in the computing environment. [10] This means, according to their findings, that a single vulnerability may be classified differently by different taxonomies depending on what level of abstraction was being observed. They proposed the concept of looking at the vulnerability at all levels of abstraction to provide a more complete and generalizable description of the problem. In a sense, they did not build a new taxonomy of vulnerabilities, but merely tried to present a new way of looking at the problem. Our work in visualizing the vulnerability windows of a layered solution follows along those same lines. We are not proposing a way to determine whether or not a layered solution is an effective strategy, nor predicting how secure one will be in the future, but offering a different way of looking at the problem, by way of a case study. For these reasons, this is the conference for which we believe our paper is the best fit, and where we would endeavor to present it.

**Broader Impact**

Being able to re-assess the ability of an individual mechanism, or “layer”, in a layered security solution in maintaining adequateprotection to both the system it is tasked with protecting and the other layers that help provide that protection is key when attempting to evaluate the effectiveness of that system. This would allow owners of the protected system as well as the providers of said system a chance to review its effectiveness and to take the appropriate measures in bolstering defenses or preventing breaches. Because a majority of the major breaches that occur today take advantage of older vulnerabilities that exist in unpatched systems, having this ability to view areas that could potentially have been vulnerable at one point is a major asset.

Though this project is not an analysis of specific vulnerabilities the fact that it takes into account as many known vulnerabilities as possible is an interesting aspect. It is an analysis of the trends of these layers and their vulnerabilities over time for the purpose of helping better those layer’s security. This benefits nearly everyone from public to private sector as well as academia. It allows a more secure environment as well as helping further the analysis of these security solutions.

Overall, the end consumer, who may not be of a technical mindset, is more secure. Allowing the developer of the solution the ability to more concretely look back at their system and use that information and those discovered past trends to further secure development of their product is a benefit for all involved.