International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

SMART: Security Model Adversarial Risk-based Tool

Pavan Navandar

Cybersecurity SAP Security Engineer, TCS, USA

ABSTRACT: As development and deployment of secure systems continue to grow at scale, there is an equal need to evaluate these systems for vulnerabilities and other problems. However, the process of evaluating these designs is complicated and mainly proprietary to the group performing the evaluation. Generally, one follows the generic risk equation of probability and impact. In addition, one should examine the costs related to the adversary and the defender of a system. Without accounting for all of these different aspects, one cannot expect to properly assess the security of a system model or design. This work presents a security model adversarial risk-based tool (SMART) for systems security design evaluation. Our tool reads in a systems security model an attack graph and collects the necessary information for the purpose of determining the best solution based on a calculated security risk represented as a monetary amount. The advantage of the tool is the level of automation provided in the evaluation of security attack trees while providing meaningful metrics that are effortless to compare.

KEYWORDS: SMART, Adversarial Risk Modeling, Attack Graphs, CVE Analysis, Security Risk Assessment, Cybersecurity Metrics, CVSS Scoring, System Security Design

I. INTRODUCTION

Systems design evaluation is done from a series of different perspectives and angles – from the cost of a system to its effectiveness/coverage. As cybersecurity concerns grow there is an equally large need to calculate security risk (SR) for these same systems. There are a series of complications that can arise when determining SR for any network/system. The type of attack being performed against a model (e.g. shotgun, direct) determines the variations of attack vectors that a malicious individual may use. One of the largest, and more difficult, factors for establishing the SR of a design comes from ascertaining the capabilities and motivations of an attacker. This can influence how effective an attacker is at exploiting a given vulnerability within a design model. The last, and possibly the most difficult, value to verify is the costs of performing an attack. The most complicated aspect of this calculation is that the steps taken to exploit a vulnerability can greatly differ based on the attacker's skill, knowledge, insight, and "divine inspiration." While there is some work that can be done to estimate these values, as shown in the work by Bayoumy [1], this also requires some knowledge on the subject for proper evaluation. Additional costs can come from the type of equipment used and the time spent in research, development, and execution.

In an ideal situation, one would have a tool that is provided a design model and produces the corresponding SR for the model through analysis of the corresponding attack graph for the model. The design model should be able to be provided in any modeling language (e.g. AADL, UML, SysML). The tool should then extract an attack tree from the provided design. Next, the attack tree's common vulnerability exposures (CVE) and non-CVE vulnerability information should be collected. Lastly, it should produce the SR for the design based on the generated attack tree. A risk measure when expressed as a monetary value provides a method of comparison between models in a meaningful way. Moreover, this monetary risk value can be incorporated into an evaluation of a system's lifetime costs. However, due to limitations in automating the process, the desired flow of a SR tool would currently require a lot more user interaction. Given a certain system model, the user identifies any assets of concern and the values associated with them. From here the user will generate an attack tree manually for each of those assets. Assets could include data as well as access to services or physical devices. Attacks on these assets could be unauthorized access, denial of access/service, destruction of assets, etc. One could construct a taxonomy or classification of these assets and attacks, but that is beyond the scope of this article. The estimation/evaluation of value for any given asset, especially establishing security-based ones, has a fluctuating subjective weight to it. Additional complexity comes when including the basis of one's evaluation (e.g. industry, academia, financial institution, service provider, malicious actor). For example, unauthorized access to private data may have different costs to a bank as compared to a hospital. The cost of performing these evaluations is extensive in both time and effort [2, 3], but even these costs carry a subjective element relative to a secluded market.

Our developed security model adversarial risk-based tool (SMART) for systems security model and design evaluation handles the interpretation of an attack graph into a SR value. First, the tool reads in an attack graph where each root node is an asset of importance to the user. Each attack graph is then pulled apart to examine the assets, vulnerabilities, and elements. For any CVE-known vulnerabilities, the SMART tool looks up to the CVE information (CVSS v2 score) attached to the vulnerability. This is the value that is then scaled and used for the probability of success for an attacker to pull off an exploit. For other non-CVE vulnerabilities, the tool requests the probability information directly from the user. Finally, SMART calculates the SR value based on the collected information. One should notice that the majority of inputs to the model are cost-based and therefore leads the SR-value representing a monetary value (e.g. USD). At this final stage, a user can then compare different model designs in a meaningful and comparable manner. The notable advantage here is that unlike most security-oriented watermarks, SMART presents straightforward data whose meaning is accessible across a large audience.

In prior work [4], we introduced a new meaningful security and risk-based metrics that take into account asset values for both attacking and defending parties. The mathematical model allows for the contrast and comparison of various system security designs and models. Since the equations are derived from a traditional risk equation, SMART is able to incorporate the monetary values and costs of elements into the final result. This model, described in further detail in 'Calculating SR' section, acts as a starting point for the development of this article's adversarial risk-based tool described in 'Attack Graph Security Risk Tool' section. The amalgamation of automation, known as security metrics, and user input presents an efficient and repeatable method for producing monetary security metrics for disparate/diversified model designs. The tool is demonstrated with an example in Section 4.

II. RELATED WORK

Building on a traditional risk model, work has been done by Armando and Compagna, Brucker *et al*., and Gonzalez *et al.* to examine the causes of risk as well as different ways to model these concerns [5–7]. Others have examined what works for modeling and understanding SR as well as means to improve current techniques [8] and [9]. Further SR research has been made into determining the best way to present risk factors [10, 11] to allow better comprehension and representation. Shameli-Sendi *et al.* [12] produced a taxonomy of risk assessment techniques for the purpose of better understanding risk assessment and for aiding in conducting proper risk assessment. Chockalingam *et al.* [13] conducted a systematic literature review to identify key characteristics and applications of safety and SR assessment. Kong *et al.* [14] pursued the development of a framework for presenting SR assessment of smart cars. Wangen *et al.* [15] surveyed various methods for information security risk assessment (ISRA) to determine the completeness of each as well as analyzing gaps in them.

Bayoumy's work [1] illustrates the background research required in order to examine the DarkNet market place to determine the cost/price of ransomware. This type of information is paramount for estimating attacker costs. However, this data requires observations over economies and reflections of social interactions that are involved in the creation and distribution of ransomware. While not a perfect solution, this type of work allows for an improved assessment of adversarial-based values. Bernsmed *et al.* [16] generated a model for producing an arbitrary value for representing risk of an unwanted event. Unfortunately, this information is still difficult to compare to other systems of risk analysis but can be used relatively in conjunction with other values. While there is a large body of work in the area of SR assessment and evaluation, there is no current standard methodology of risk calculation. Our work moves to help solidify best practice for determining monetary value for adversarial risk-based evaluation of security system design and modeling.

Attack graph security risk tool

In an ideal case one would have an automated design of attack graphs, but for the sake of simplicity we simply go over the different aspects of the current developed SMART code, as well as exemplification of each piece examining a given attack graph. Normally one would create a separate attack graph for each asset of importance to the user. The "Asset of Importance" (AoI) is the key element or data that a user is interested in protecting from malicious access, alteration, or harm. When creating an attack graph for each asset, one should follow a few qualifiers. All other items in the attack graph are not seen as that graph's asset of importance. If an asset of importance exists within the "path" of another asset of importance, then one should create a separate attack graph for that asset. When evaluating an attack tree with embedded assets it does not make sense to re-copy the attack trees since each asset would give you the value of the tree up to that time. Therefore, one would really only need to analyze the tree multiple times to determine the overall SR of the design model. The value to the attacker (e.g. '*A*'-value) for all other nodes is seen as zero (0); which can be ignored for the sake of the calculation. The reason for this being that the malicious individual is solely interested in the asset of importance and not any of the other elements.

A visualization of the SMART is shown in Fig. 1. Our developed tool (SMART) works in roughly the following three stages:

1. The user produces either one or multiple attack graphs based on an original system design model that requires evaluation. The tool then reads in each user-supplied attack graph for a given systems security model. This provides a unified manner of presenting information so that various models can be presented to the same tool; delivering a level of automation.
2. The tool then begins to collect the necessary information to account for defensive costs as well as adversarial-based risk factors. An Application Programming Interface (API)-based collection of CVE information for known vulnerabilities is referenced, as well as the collection of other non-CVE vulnerability information and element-specific variables from the user-maintained parameter database. The automated portion of this step adapts a well-known value (CVE) as a risk approximation while also taking in a known number of user-determined values (e.g. cost, personal value) which allows for a straightforward and repeatable process across multiple evaluation runs.
3. Finally, SMART performs the calculation of SR for the provided model based on the collected information. The SR of each potential attack path is computed, followed by the aggregation of overall design's SR cost. This value is presented as a monetary amount (USD). On evaluating to a monetary value, this allows for the tool's output to be understood and easily compared by a wide audience.

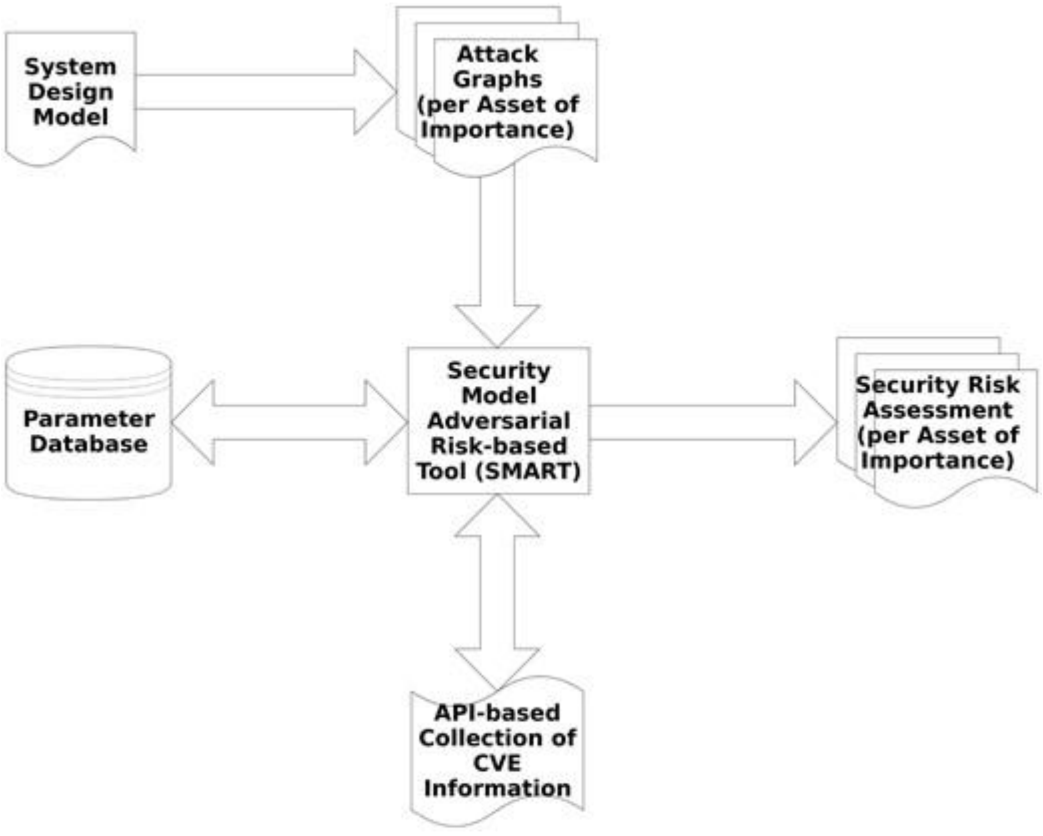
System Design Model
Attack Graphs (per Asset of Importance)
Parameter Database
Security Model Adversarial Risk-based Tool (SMART)
Security Risk Assessment (per Asset of Importance)
API-based Collection of CVE Information

Figure 1: Illustration of flow graph for SMART.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

Altogether, the tool not only accounts for all the necessary costs/values associated with the attack graph, but also takes into account each node and attack path for accurate calculation of the system security model's overall SR. SMART automates the process of evaluation while making it easier to understand and compare its output. While SMART is robust at a smaller scale, it is modular enough to easily adapt to examinations of larger and more complex spaces. In addition, one should note that the additional costs (e.g. attacker value, miscellaneous black-market costs, time) would need to be verified externally as this tool assumes that the values are acceptable to the user. Since these additional evaluations can be extremely personal/secretive, it is understandable why individuals/groups would not want to openly share this information. While these aspects are of great importance, their determination to the user is considered out of scope for this work. SMART allows for automation for evaluating a variety of designs for their SR in a widely recognized monetary value (USD). It also incorporates user-defined values that can be shared and altered between multiple designs. By presenting a flexible and reproducible manner of evaluating these models, our work helps to automate this assessment step in the larger design process.

Attack graph

Attack graph generation can be automated through tool-based interpretation of a provided system model. We build our attack generation on the AASPE attack graph tool for AADL models [17]. Development and evaluation of security models through the use of attack graphs have been delved into via a plethora manners and is still a relatively new field. Exploration of this space involves building on the previous and emerging techniques, methods, and processes for generating and interpreting SR/costs through the use of attack/defensive graphs [18, 19]. In these cases one must determine the most effective way to present elements, costs, impacts, and probabilities [20, 21]. In addition, one must determine how best to represent actions and values to attacking/defending parties. Evaluation of these aspects present further techniques to characterize economic value and worth of each. For our work we build upon the mathematical foundations we established in [4] and combine this with the previously discussed AASPE tool set's attack graph model as input to SMART. Use of the AADL attack graph model was chosen due to its simplicity and customization for user input. In addition, AADL allows for more visual development, ease of interaction with user tools, and ease of learning curve for new users. Therefore, this was a prime candidate for preparing an example attack graph to build SMART upon. For the purpose of this work, we use attack graph shown in Fig. 2 as an example.

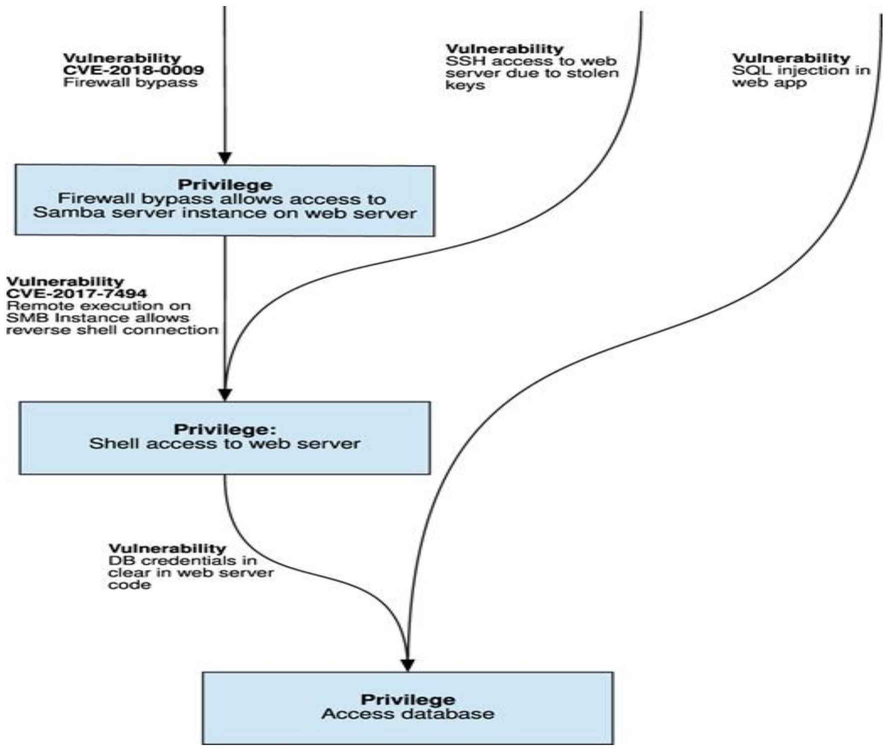
Vulnerability
CVE-2018-0009
Firewall bypass
Vulnerability
SSH access to web server due to stolen keys
Vulnerability
SQL injection in web app
Privilege
Firewall bypass allows access to Samba server instance on web server
Vulnerability
CVE-2017-7494
Remote execution on SMB Instance allows reverse shell connection
Privilege:
Shell access to web server
Vulnerability
DB credentials in clear in web server code
Privilege
Access database

Figure 2: Example of an attack graph for the single firewall network architecture.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

Figure 2 shows a subset of the attack graph for the single firewall architecture. Note that a full attack graph would be significantly more complex. Each node in the associated attack graph represents an escalated privilege enabled by a vulnerability designated on an incoming arc. Vulnerabilities can be extracted from the CVE database if the device is known, or from penetration testing. In our example graph, we show three attack vectors to a target privilege asset which is the internal database. On the left is a path that exploits a vulnerability that allows a specific firewall to be bypassed thereby enabling access to an SMB service on the web server that was supposed to be blocked. Once connected to the SMB server, the attacker can exploit a vulnerability in Samba that allows remote execution which can lead to reverse shell access to the web server. In this case, we assume that credentials to the database are written in cleat-text within the code, and the attacker can then connect to the database. Other possible vectors of attack are direct connection to the web server because of stolen SSH keys or direct connection to the database because of SQL injection in web server application code. Obviously, these three attack vectors are only a small subset of the possible attack paths to the database.

The attack graph is represented as an XML tree with a “Root Node” with different levels of indentation for additional “subNodes” and corresponding vulnerabilities. Nesting of the vulnerability and subNode information can continue until such a point that the entirety of the asset’s attack graph has been embodied. Within each node and vulnerability tag there are fields that contain the important information that SMART will use and interpret. These fields are “name” and “description.” Inside of the subNodes tag these fields are used to provide the name of each element in the attack path as well as a description of that component. When used in tandem with the vulnerabilities tag, the name field represents either the CVE associated with a known vulnerability or used as a label for a more general security concern (e.g. SQL injection). The description field is used to provide additional detailed information about the known vulnerability. Once the attack graph XML representation has been created, this model is fed into SMART to begin the automated collection of SR information.

CVE API

Our adversarial risk-based tool first will read the attack graph provided to begin determining which information is present within the graph, what data will need to be queried for, as well as any information that must be obtained from the user and/or parameter database. The aspect that will be examined in this Section relates to the collection of CVE data. As SMART reads each level of the XML attack graph, CVE information is collected and processed based on the shape and form of the various nodes and connecting edges. SMART makes use of the cve-search tool public search operated by CIRCL [22, 23]. Through this API, our tool collects the CVSS V2 information for any provided CVEs. This value functions as a representation of the probability of success for an attack upon that vulnerability. It should be noted that the reason that the CVSS V2 score is used is mainly due to the current API being used to collect the CVE information. It is also worth noting that there is concern around the use of CVSS scores. One such problem is the seemly arbitrary way that the V2 and V3 scores can differ so disparately for the same vulnerability. However, despite this concern of “stability” the core reason behind the use of these scores is that they represent an accepted scale across all the known CVEs.

Calculating SR

Once SMART has collected all the necessary information from the user, along with the necessary CVE CVSS score data, our tool moves to produce the appropriate SR value. The security model adversarial risk-based tool considers each node within the attack tree graph and examines the different attack and vulnerabilities of the node itself as well as the edges connecting to neighboring nodes. Contingent on the attack path(s) being examined, different forms of calculation are used to produce the overall SR of a provided model. These two types of computation are either a summation, or a product of the values present

The more prevalent of the two is summation. The first level of summation performed is a representation of the aggregation of all contributing attack vectors toward the potential compromise of a specific system asset. This is shown by Equation (7). The second aggregating scenario is a final summation of all attack vectors P in an attack graph G_i for a particular asset p. Equation (2) shows this final form of the SR calculation done by SMART. To account for the effect of multiple potential exploits along the attack trees branches, we implement a product of values. The overall probability of success for any attack path along the graph model is calculated by taking the product of the probabilities of successfully exploiting each vulnerability edge. This relationship is exemplified by Equation (1), where v is a vulnerability edge on path P and is the probability of successfully exploiting v.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

Our SR model is summarized here, but further details on the model are available in [4]. We start with the traditional risk model as defined below:
We modified the traditional risk model to represent the security risk, SR, as a combination of the probability of an attack (p_a), probability of the attack succeeding (p_s), and the impact of the attack

The probability of an attack (p_a) is defined as the chance that a given system, or element, is going to be targeted by an attacker's efforts. In other words, if the system is not of any interest to an attacker, i.e. $p_a = 0$, the risk is 0. If the system is highly desirable to attackers, i.e. $p_a = 1$, the risk is dependent completely on the probability of success (p_s). p_s is defined as the chance that an attacker's efforts will pay off positively for the attack, or negatively for the defending system. The probability of success is thus tied to the quality of the security defenses built into the system design. Later, we discuss a methodology to calculate p_s based on an analysis of attack graphs. However, an initial approximation can be represented by the CVSS value discussed earlier.

Now, we turn our attention to determining an appropriate value for p_a. First, we introduce a new variable (A) which defines the value of an attack to an adversary. This is distinct from I, which is the impact of an attack on the defender. As an example, a credit card number has a current value on the dark web of a few dollars to an attacker. However, a defender may have impact on costs of hundreds of dollars due to lost reputation, credit monitoring fees, regulatory fines, etc. Similarly, different malicious individuals may also assign different values for A to assets under attack. For our purposes, we assume a single attacker model, but one could arrive at an aggregate A with weighted assignments due to various adversaries. We note that as the expected attack value, changes, so should the probability of an attack (p_a). In other words, as the expected value of an attack increases, the likelihood of an attack will also increase. It is also worth noting that p_a is influenced by the cost to attack (c_a) a given system, in such a manner that if then $p_a = 0$; otherwise, if then p_a should grow from 0 to eventually 1. This is because p_a should be zero until such a time that the expected value, , is greater than the cost to attack. Other than these broad guidelines, the exact relationship between the expected attack value and p_a is not well defined and further study is needed to determine an accurate model. As a simple model, we suggest an exponential relationship as shown where α captures the sensitivity of p_a to the expected attack value. Determination of appropriate values of α is dependent on the particulars of an attack scenario. On incorporating into Equation (4), the equation for determining SR becomes:

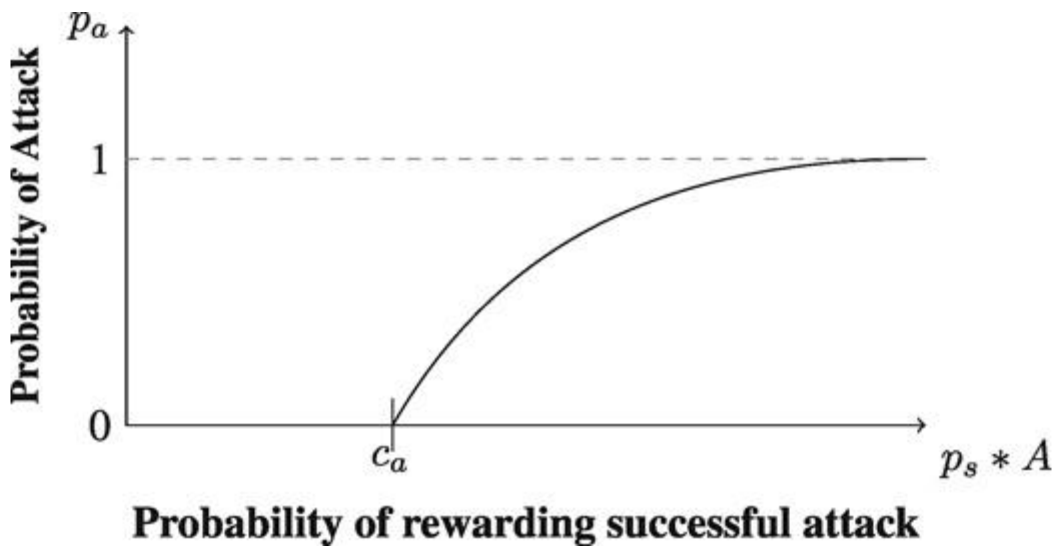

Figure 3: Estimation of attack risk behavior.

Thus, resents the aggregation of all contributing attack vectors toward the potential compromise of a specific system asset. This SR metric, essentially, represents the expected security cost of an element over a lifetime. The metric captures the level of system SR influenced by both the potential of attack as well as impacted loss. This means that the SR of a given solution is influenced by both the monetary value that an attacker places upon the presented system as well as what is required by the defender to recover from a successful attack. Using the resulting SR metric, a developer can tailor their design to better protect systems while also presenting a higher cost to an adversary willing to expend resources. The *SR* value allows for the approximation of a monetary value of the security strength of any potential design solution. However, while the calculated metric does begin to influence implementation decisions, it still requires incorporation into the overall cost of acquisition, operation, and maintenance.

User parameterization

SMART does assume that there are certain pieces of information that must be provided by the user. The reason being that this level of data could be proprietary or unique to the scenario being examined. Examples of this type of information include: impact of compromised asset (I), value of an asset to an attacker (A), cost of attacking a given element (c_a), cost of implementing/maintaining/operating a specific system component ($c_i/c_m/c_o$). In a scenario where SMART does not have all the data it requires, the tool requests any missing information from the user, or a user-supplied parameter database. One should note that this does require the availability of a “security guru” who would have all necessary information on hand. This is seen as an acceptable assumption at this stage because without knowing at least an estimation of these values, we can assume that a SMART-level evaluation would not be occurring. As stated in ‘CVE API’ section, depending on the relation of the various nodes and edges in the original attack graph, different equations are used to calculate the overall SR. These scenarios are further detailed in the following section.

Design evaluation

In review, the SMART tool currently requires that an individual manually create an attack graph representation of a model that includes all the necessary CVE and other vulnerability information. Once the XML design is ready, this attack graph representation is fed into SMART along with any costs associated with elements in the model as well as any probabilities of success for non-CVE vulnerabilities present. From this data our tool produces an overall SR as a monetary value that is associated with the asset of interest that can be compared in a meaningful way. It is through this final information that one can effectively see the difference in risk between one security model and any other differing design.

For illustrating the use of SMART, we move to show how three different choices in design architecture can influence the overall risk of a system. To a simplified example, we focus on a defensive architecture choice that protects against the denial of service (DoS) attack. In this scenario we examine a simple database server being protected by a firewall, shown in Fig. 4.

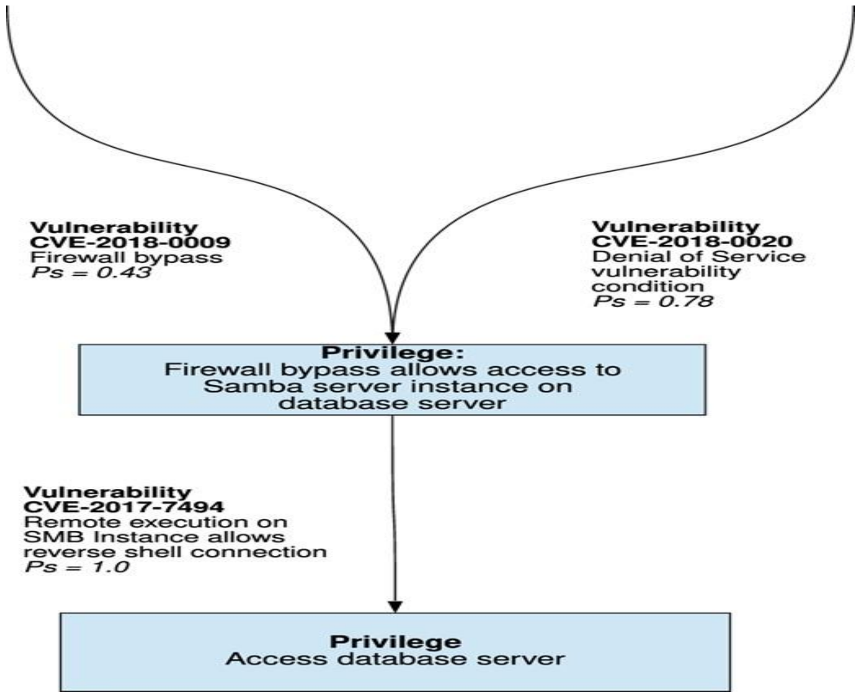
Vulnerability
CVE-2018-0009
Firewall bypass
Ps = 0.43
Vulnerability
CVE-2018-0020
Denial of Service vulnerability condition
Ps = 0.78
Privilege:
Firewall bypass allows access to Samba server instance on database server
Vulnerability
CVE-2017-7494
Remote execution on SMB Instance allows reverse shell connection
Ps = 1.0
Privilege
Access database server

Figure 4: Example of an attack graph for the simple firewall network architecture.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

The difference in design that will be examined is the inclusion and then removal of the DoS vulnerability from the firewall element through additional DoS detection. SMART is then run using the three differing attack graphs to produce each design's corresponding SR value. For the purpose of showing the effect of different element's security rating, we also include a model where the non-DoS vulnerability is addressed but the DoS vulnerability is not. This does mean we are only looking at a change in vulnerability at the leaf of one branch of the overall security model. It is worth noting that depending on where the increase/decrease of potential vulnerabilities exist, the overall design calculation will be affected differently.

The main effect that one should notice here is that by implementing a firewall that can detect the presence of a DoS attack, one is able to nullify the associated risk of known DoS vulnerabilities. This, in turn, changes the overall probability of successful attack upon the security model, improving the overall risk to the system, and producing a lesser SR value. The results of this calculation are shown in Table 1. The contents of the table reflect the values for different paths/elements (e.g. "[]") as well as the costs associated with the asset of importance (e.g. Database server), for Figs 4, 5, and 6. It should be noted the values for the asset A, paths c_a, and asset I are scaled by 1000.

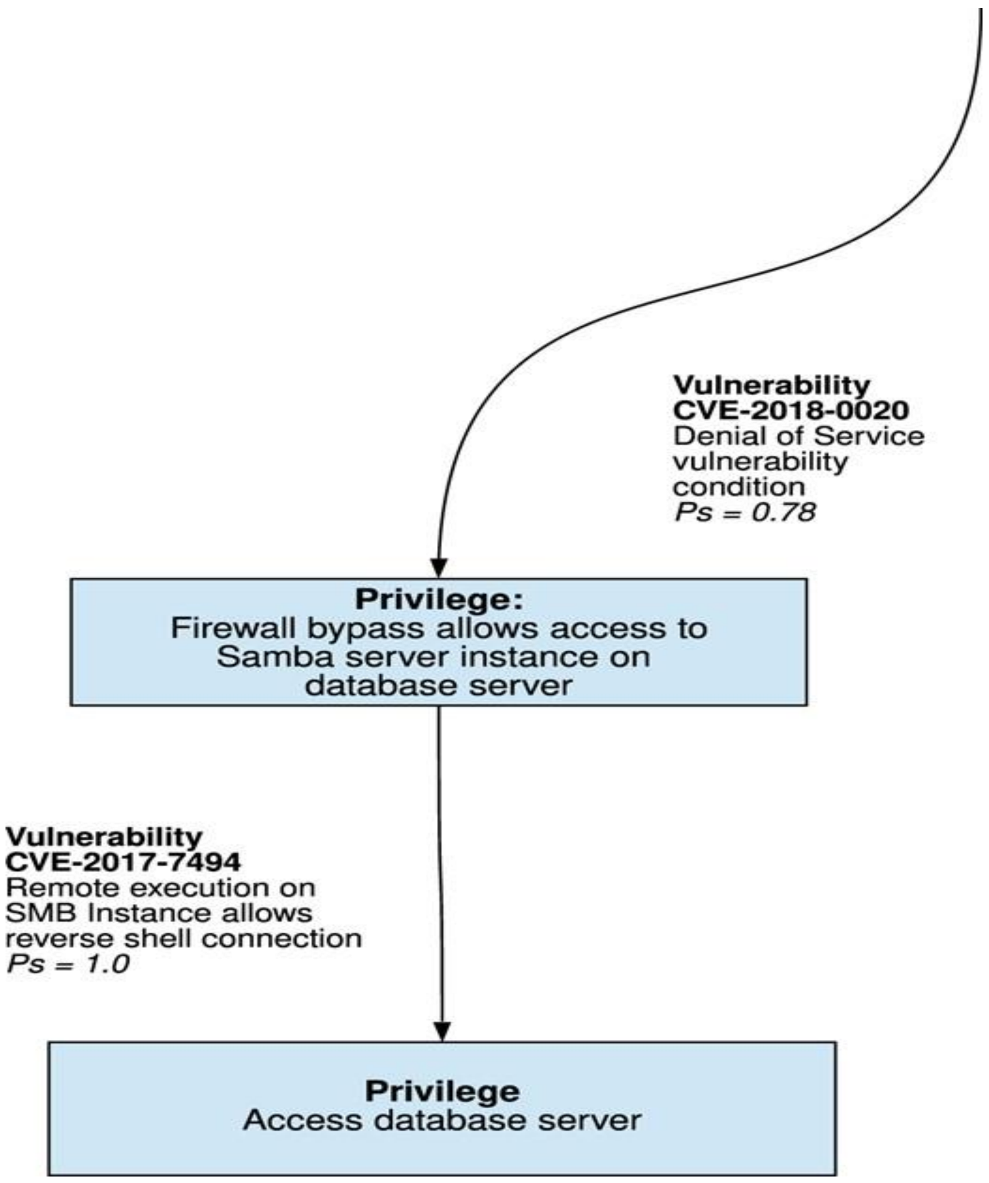
Vulnerability
CVE-2018-0020
Denial of Service vulnerability condition
Ps = 0.78
Privilege:
Firewall bypass allows access to Samba server instance on database server
Vulnerability
CVE-2017-7494
Remote execution on SMB Instance allows reverse shell connection
Ps = 1.0
Privilege
Access database server

Figure 5.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

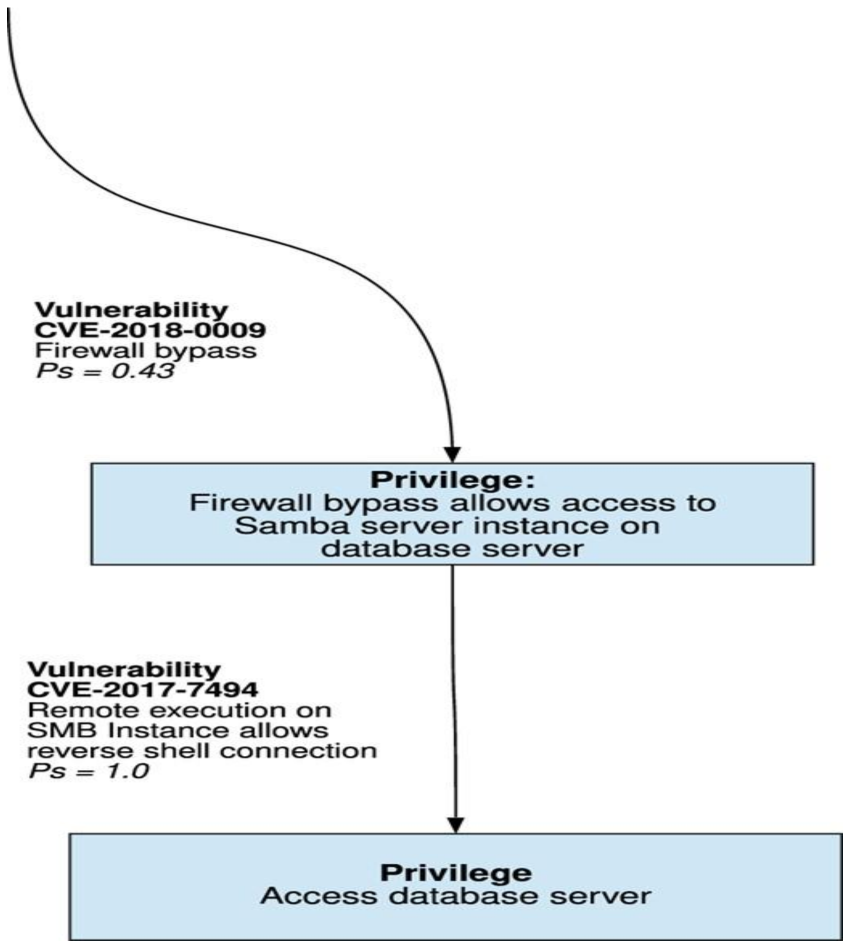

Figure 6.

Table of values for calculating SR using Figs 4, 5, and 6. Each column is a different variation of the network architecture

	Graph paths p_s	**Asset attack value (scaled USD)**	**Graph paths c_a (scaled USD)**	**Asset impact (scaled USD)**	**Graph paths c_i (USD)**	**Calculated SR (USD)**
Simple DoS	[0.43, 0.78]	20	[8.0, 5.5]	11.7	[8000.0, 10000.0]	$101 845.55
Simple DoS edit	[0.78]	20	[5.5]	11.7	[8000.0, 10000.0]	$99 575.63
Simple DoS fixed	[0.43]	20	[8.0]	11.7	[8000.0, 10000.0]	92 719.93

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

The values for "Cost of Attack" (c_a), "Cost of Initialization" (c_i), "Impact" (I), and "Attacker Value" (A) were estimated for the purpose of this investigation. In an ideal case, this information would come from internal analysis of systems. To improve SMART, we developed a series of JSON-based parameter databases containing the "standardized" values for various assets, elements, and vulnerabilities. For the purpose of simplifying the calculations, and to show the use of the SMART tool, we use the same Cost of Operation (c_o) and Cost of Maintenance (c_m) for all elements; \$2450 and \$70 000, respectively. This allows for a clearer example of how the tool's output is affected by the vulnerability and element details.

As shown in Fig. 7, the effect of Impact (I) and Attack Value (A) can be seen on the overall SR. As one can see, the A value is not as important to the overall SR as the I value. It is also worth noticing that there is a "fold" that exists within the graph that indicates a sharper jump in overall *SR* cost. This fold is caused by the conditional related to c_a in Equation (5). When the attack value, A, passes the threshold, the attack becomes more attractive and the SR increases. The main point to take away from Fig. 7 is that while the lion's share of risk comes from the potential loss/impact, there is a noticeable influence to risk based on the value of an asset to an attacker. The effect of increase in impact makes sense, since this aspect is treated as the basis of risk calculation, e.g. the greater the impact the greater the SR. What is interesting is to see how the increase in attack value influences a change to the overall *SR* value, due to the greater effort an attacker is willing to put into an attack. In other words, as the attacker places greater value (i.e. becomes more interested in a target/asset) the SR of the system increases. This is best illustrated in the figure where one can see a sharper increase in SR where the attack value is highest.

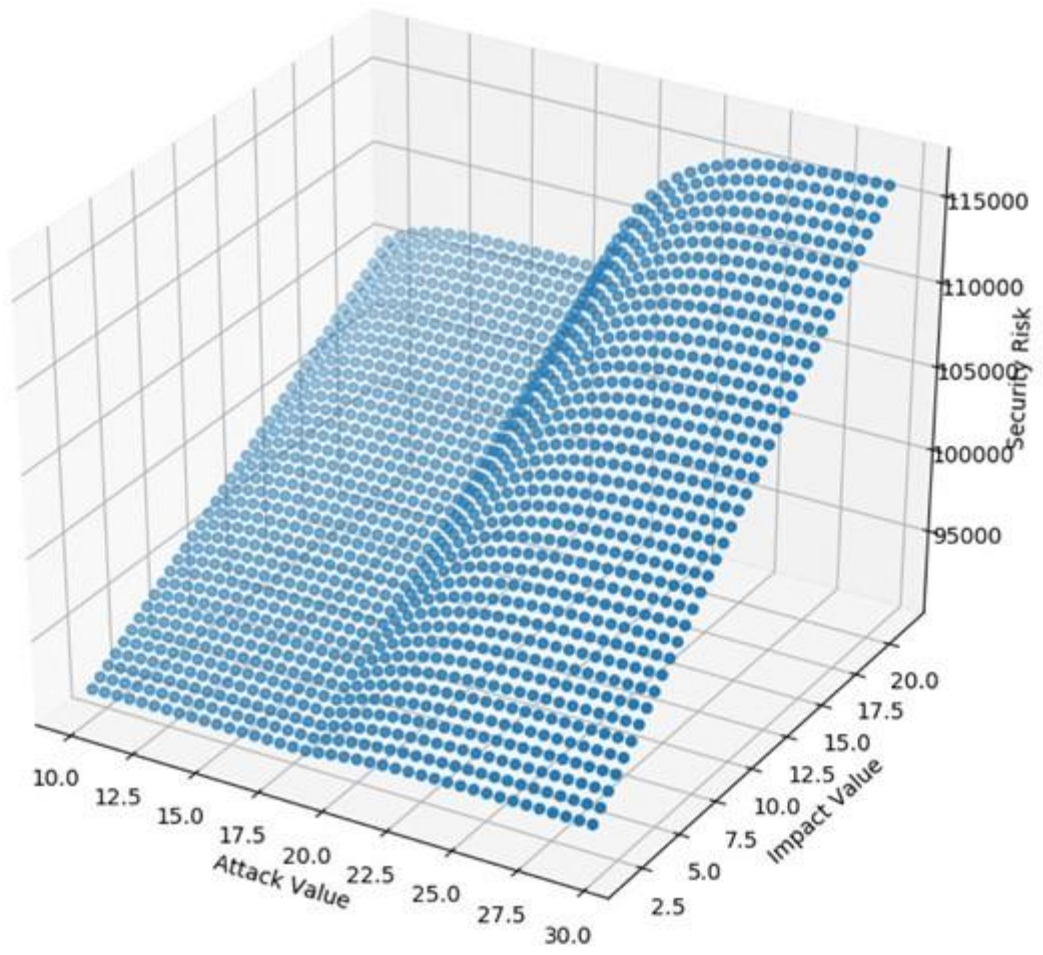

Figure 7: Illustration of impact and attack value effect on SR.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

III. CONCLUSION AND FUTURE WORK

We have introduced SMART for systems security model and design evaluation. Our tool allows a user to generate an XML-based attack graph and feed this graph model into SMART to produce monetary metrics of SR that can be compared in a meaningful way. By building upon earlier techniques and mathematical relationships, we have managed to automate the comparison of system design models at a SR level. Through our work, we present an easily repeatable process for comparing and evaluating the SR of design models. This allows for a standardized way to obtain a monetary-based and comparable value. Incorporating this tool within a larger security design framework would allow for catching dangerous system/model configurations and implementations before such work could get to an implementation stage.

Future work includes automating the generation of attack graphs based on the original design, improving collection and storage of values to allow easier interaction with the user of SMART, and incorporate pruning of the generated attack tree. Furthermore, effort needs to be put into the identification of potential vulnerabilities (both CVE and non-CVE) based on specific attack graph elements. In this manner, the tool will be able to better determine "default" values for these vulnerabilities, costs, impacts, and attacker values.

REFERENCES

1. **Yin, Z., Jain, M., et al.** (2012). Game-theoretic resource allocation for malicious packet detection in computer networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Richland, SC: IFAAMAS.
2. **Durkota, K., Lisy, V., Kiekintveld, C., et al.** (2015). Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*. Richland, SC: IFAAMAS.
3. **Blocki, J., Christin, N., Datta, A., et al.** (2013). Audit games. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*.
4. **Blocki, J., Christin, N., Datta, A., et al.** (2015). Audit games with multiple defender resources. In *AAAI Conference on Artificial Intelligence (AAAI)*. Palo Alto, CA: AAAI Press.
5. **von Stackelberg, H.** (1934). *Marktform und Gleichgewicht*. Vienna: Springer.
6. **Kiekintveld, C., Jain, M., Tsai, J., et al.** (2009). Computing optimal randomized resource allocations for massive security games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 689–696. Richland, SC: IFAAMAS.
7. **Leitmann, G.** (1978). On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications*, 26, 637–643.
8. Navandar, Pavan. "Fortifying cybersecurity in Healthcare ERP systems: unveiling challenges, proposing
9. Navandar, Pavan. " Enhancing Security with Two-Factor Authentication in SAP Fiori Applications" Journal of Scientific and Engineering Research 5, no. 10 (2018):329-33.
10. Navandar, Pavan. " Segregation of Duties (SoD) Risks in SAP Security: Mitigation Strategies and Best Practices" Journal of Scientific and Engineering Research 6, no. 9 (2019):206-206.
11. Navandar, Pavan. " SAP Security is key for Business Success for ERP system" Journal of Scientific and Engineering Research 5, no. 6 (2018):398-400.
12. **Breton, M., Alg, A., & Haurie, A.** (1988). Sequential Stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59, 71–97.
13. **Conitzer, V., & Sandholm, T.** (2006). Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, 82–90.
14. **Paruchuri, P., Pearce, J. P., Marecki, J., et al.** (2008). Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *Proceedings of the 7th International Conference on*
15. *Autonomous Agents and Multiagent Systems (AAMAS)*, 895–902. Richland, SC: IFAAMAS.
16. Navandar, Pavan. "Enhancing Cybersecurity in Airline Operations through ERP Integration: A Comprehensive Approach." Journal of Scientific and Engineering Research 5, no. 4 (2018): 457-462.
17. Navandar, Pavan. " Enhancing Governance, Risk, and Compliance (GRC)" Journal of Scientific and Engineering Research 7, no. 3 (2020):250-256.
18. Navandar, P. (2021). Fortifying cybersecurity in Healthcare ERP systems: unveiling challenges, proposing solutions, and envisioning future perspectives. Int J Sci Res, 10(5), 1322-1325.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 5, Issue 2, March – April 2022||

DOI:10.15662/IJRAI.2022.0502003

19. **Korzhyk, D., Conitzer, V., & Parr, R.** (2010). Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 805–810.
20. **Yin, Z., Jain, M., Tambe, M., et al.** (2011). Risk-averse strategies for security games with execution and observational uncertainty. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 758–763.
21. Navandar, P. (2021). "Developing Advanced Fraud Prevention Techniques using Data Analytics and ERP Systems" Int J Sci Res, 10(5), 1326-1329.
22. **An, B., Tambe, M., Ordonez, F., et al.** (2011). Refinement of strong Stackelberg equilibria in security games. In *Proceedings of the 25th Conference on Artificial Intelligence*, 587–593.
23. **Pita, J., John, R., Maheswaran, R., et al.** (2012). A robust approach to addressing human adversaries in security games. In *European Conference on Artificial Intelligence (ECAI)*. Amsterdam: IOS Press.
24. **Jain, M., Kardes, E., Kiekintveld, C., et al.** (2010). Security games with arbitrary schedules: A branch-and-price approach. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 792–797.
25. P. Navandar, "Optimizing SAP roles for efficient enterprise resource planning," Int. J. Sci. Res. (IJSR), vol. 9, no. 1, pp. 1932–1934, Jan. 2020, doi: 10.21275/SR24529194621.
26. P. Navandar, " Mitigating Financial Fraud in Retail through ERP System Controls" Int. J. Sci. Res. (IJSR), vol. 9, no. 4, pp. 1823–1827,
27. Navandar, Pavan. " Unveiling the Power of Data Masking: Safeguarding Sensitive Information in the Digital Age" International Journal of Core Engineering & Management 5, no.6 (2019): 27-32.