Vulnerability Analysis for Security Resource Allocation Problem via Penetration Testing

Xinghcen Wang
wang2930@purdue.edu
Purdue University
West Lafayette, IN, USA

Mustafa Abdallah
mabdall@iu.edu
Indiana University-Purdue University Indianapolis
Indianapolis, IN, USA

Saurabh Bagchi
sbagchi@purdue.edu
Purdue University
West Lafayette, IN, USA

ABSTRACT

We perform penetration testing to study the relationship between the safety of the targets and the security investments in cyber-physical systems (CPS) to security investments. This sensitivity captures the relationship between the probability of successful attack and the security investments on these parts within CPS. In this context, we use two intrusion detection systems (IDS); Snort and Suricata to investigate such a relationship. We implement a variety of security investments by changing the number of rules applied within the configuration of the IDS. Our experiments show that the probability of successful attack decreases exponentially with the security investments made by the security defenders. Also, we find that the sensitivity to investments (i.e., the decrease of the probability of successful attack with each additional unit of security investment) varies in different parts of the CPS.

KEYWORDS

Penetration testing, Risk Assessment, Resource Allocation, Intrusion Detection Systems , Cyber-Physical Systems

ACM Reference Format:
Xinghcen Wang, Mustafa Abdallah, and Saurabh Bagchi. 2022. Vulnerability Analysis for Security Resource Allocation Problem via Penetration Testing. In *Proceedings of ACM (WRIT'22).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Motivation

Facing modern and future challenges in control and monitoring, cyber-physical systems (CPS) play a significant role in different applications, including smart grid [5], autonomous automobile systems [15], medical monitoring [20], smart manufacturing control systems [3], etc. Given the nature of CPS (shown in Figure 1), they face increasingly malicious attacks from external adversaries. Thus, CPS defenders need to allocate their security budgets with discretion to put obstacles in the way of cyber attacks against their critical assets [4]. In particular, enhancing security decision-making and optimal actions of such CPS defenders is crucial [24]. Thus, capturing the source for such decisions can help in enhancing the design of CPS, where such source is based on the relationship between the probability of successful attack on the assets and the deployed security investments (policies).

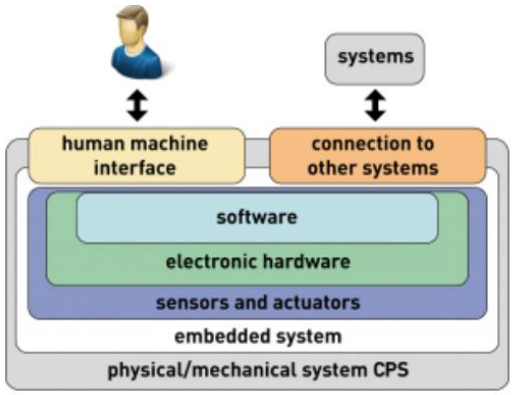

Figure 1: A typical overview of a CPS. It has physical components, software layers, and human-machine interface.

The recent works [1, 2, 4–6, 29] have studied the security of large-scale cyber-physical systems in which defenders invest their limited security budgets with the goal of thwarting the spread of cyber attacks to their critical assets. This problem of security investment allocation, made by the defenders, has been modeled as a security game. In all of these works, the theoretical analysis was based on the assumption that the probability of successful attack ideally decreases with the investment made by defenders. Therefore, it is vital to verify such theoretical assumption using practical experiments and characterize the function that employ such relationship between the probability of exploit and the security investments.

One way to practically capture such relationship is via performing penetration testing [10, 22, 26, 28]). In this context, the prior works [10, 22, 26, 28]) have showed the application of penetration testing and intrusion detection system in vulnerability analysis for security models. However, to the best of our knowledge, none of them have considered the relationship that we consider here in our current work.

WRIT'22, Dec 2022, Austin, TX, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2 Contribution

In this paper, we set up services in virtual machines that mimic services that are running in real applications and run penetration testing [8] using Metasploit [14] to estimate the relationship between the probability of successful attack and the security cost for the defender. We then verify the concept of sensitivity which is the decreasing rate of probability of successful attack due to extra security investments. In other words, for an asset with larger sensitivity values, the probability of successful attack on such asset decreases faster with each additional unit of security investment. To achieve such goals, we also leverage exponential regression [18] to find the equation to fit the relationship between the probability of successful attack and the sum of the security investments in our experiment.

1.3 Paper Organization

The rest of the paper is organized as follows. We first review the penetration testing, intrusion detection, and security model concepts that we use in our experiments in Section 2. In Section 3, we present the investigation method, the experimental setup, and the details of the implementation of our experiments. Section 4 shows our main results and findings. We discuss the relationship between insider threats and penetration testing in Section 5. We finally conclude our work in Section 6.

2 BACKGROUND

We now present a review of the penetration testing and intrusion detection system concepts that we use in our experiments.

2.1 Penetration Testing

Penetration testing [8] is the practice of testing a computer system, network or web application to find security vulnerabilities that an attacker could exploit. The security setup that we study can simply be mapped to a penetration testing to show the application of resource allocation in reducing the danger of real vulnerabilities. The vulnerabilities discovered in penetration testing can be evaluated using Common Vulnerability Scoring System (CVSS) [25]. Note that CVSS is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat level [25].

In this context, Metasploit is a popular penetration testing framework that makes such ethical hacking simple. The core Metasploit framework is both free, open-source software and comes pre-installed in Kali Linux [7] which mimics an attacker in the penetration testing setup. Metasploitable [19] is a vulnerable target which provides a secure place to perform penetration testing and security research using virtual machines. It plays as an intentionally vulnerable virtual machine for the penetration testing setup. Figure 2 shows such a process.

2.2 Intrusion Detection System

An intrusion detection system is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered [17]. A well known framework for such tasks is

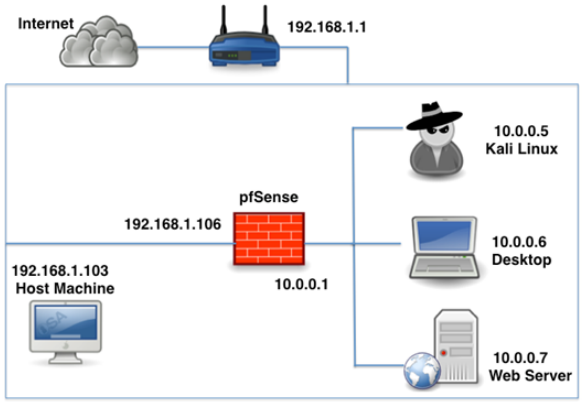

Figure 2: A high level overview of the penetration testing procedure performed in this work (adapted from [27]).

Snort [23] which is an open-source, free and lightweight network intrusion detection system software for Linux and Windows to detect emerging threats. By adding rules in the configuration of Snort, we can detect the malicious exploit sessions according to their relative Common Vulnerabilities and Exposures Identifiers (CVE ID). By installing Snort on the defender's machine of our security setup, we can decide how to allocate our security resources (or budgets) to reduce the probability of successful attack.

Similarly, Suricata [10] is also an open source-based intrusion detection system and intrusion prevention system. We replicate our experiments by performing intrusion detection along with replacing Snort by Suricata. Similar to Snort, Suricata also uses a flexible rule-based language to describe traffic that it should collect or pass. However, the rule sets of Snort and Suricata are not the same which leads to different results in our experiments. Comparing the results in Section 4, we show the generalization of our conclusions.

2.3 Security Model

We study security game consisting of one attacker and one defender interacting through an attack graph $G = (V, \mathcal{E})$. The set of nodes V of the attack graph represent the assets in the CPS, while the set of edges $\mathcal{E}$ capture the attack progression between the assets. The default probability that the attacker can successfully compromise V_d, having compromised V_a, is denoted by the edge weight $p^0_{a,d} = 1$. By the "default probability" we mean the probability of successful compromise without any security investment in protecting the assets. The defender D is in control of one asset $V_d \subseteq V$, and can make security investments on the edge $\mathcal{E} = (V_a, V_d)$. This is motivated by the fact that a CPS comprises a subnetwork owned by an independent stakeholder. Figure 3 shows such an example with a single edge (V_a, V_d).

To protect the critical assets from attacks, the defender can choose to invest his resources in strengthening the security of the edges in the network. Specifically, let $x_{a,d}$ denote the investment of defender D on edge $(V_a, V_d) \in \mathcal{E}$. In addition, let $s_{a,d} \in (0, \infty]$ denote the sensitivity of edge (V_a, V_d) to the investment $x_{a,d}$. For

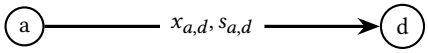

Figure 3: Attack graph instance with a single edge (V_a, V_d).

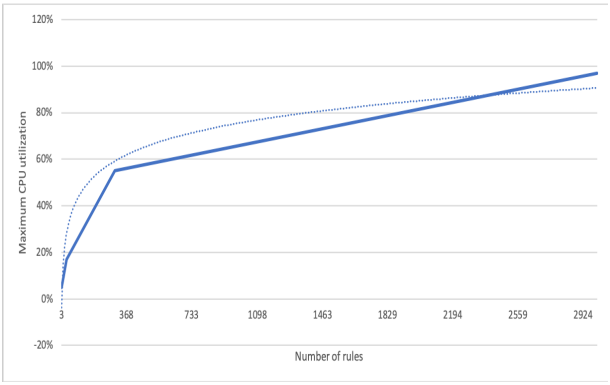

Figure 4: The relationship between maximum CPU utilization caused by Snort and number of rules applied in IDS.

larger sensitivity values, the probability of successful attack on the edge decreases faster with each additional unit of security investment on that edge. Then, the probability of successfully compromising V_d starting from V_a is given by,

$$p_{a,d}(x_{a,d}) = p^0_{a,d}\, exp(-s_{a,d} \times x_{a,d}) \quad (1)$$

Under this assumption, the probability of successful attack on an edge (V_a, V_d) decreases exponentially with the investments on that edge by defender. We will verify this assumption using practical experiments in Section 3.

Remark: The recent works [1, 2, 4–6, 29] modeled such security problem and showed several properties regarding the security resource allocation patterns on the attack graphs. These assumption was based on other seminal security literature [9, 11]. However, none of these performed vulnerability analysis via penetration testing to validate the relationship between security policies and arising probability of successful attack. On the other hand, our current work aims to consider such vulnerability analysis via simple penetration testing using both Snort and Suricata frameworks. In particular, we examine the different parameters in Equation (1).

3 METHODS

In this section, we present the experimental setup and the details of the implementation of our experiments to investigate the relationship between the probability of successful attack and the security investments, given by (1).

3.1 Experimental Setup

All experiments are based on penetration testing which is consisted of one virtual machine as attacker and another virtual machine as victim. We use VirtualBox-6.0.6 to create two virtual machine on a Mac PC. Then we install Kali-2019.2 Linux system on attacking machine, which contains Metasploit framework, and install Metasploitable-2.0.0 Linux system [26], an intentionally vulnerable Linux virtual machine, on our first victim machine. To verify the concept of the sensitivity, we also install Ubuntu-18.04.2 on our second victim machine. We also install Snort-3.0, an intrusion detection system, on both victim machines. By adding rules in configuration of Snort, the vulnerable exploit sessions would be potentially able to detect attacks and thus it should decrease the probability of successful attack. We show our experimental setup in Figure 5. For security cost, we consider the maximum CPU utilization as the cost in our experiment. In particular, we consider the worst case scenario when CPU utilization reaches maximum utilization. As Figure 4 shows, the maximum CPU utilization caused by IDS monotonically increases with number of rules applied in IDS. Therefore, we use maximum CPU utilization to illustrate the security cost.

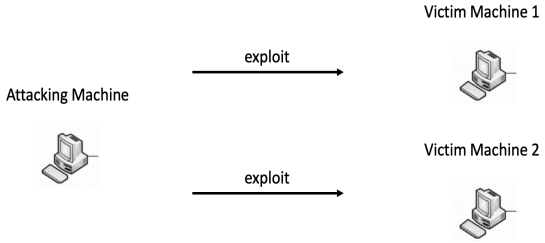

Figure 5: Expertment Setup

3.2 Experiment Step

We add "db_autopw" plugin [12] to attack machine and automatically exploit the victim machine. In [21], "db_autopwn" plugin has been recommended for automatic exploiting offered by the Metasploit framework. This plugin can find all the potential vulnerabilities in the target virtual machine. It makes the exploiting process efficient and automatic. We record the number of successfully exploiting modules as N_0. We install IDS in victim machine and configure it into intrusion detection mode. Each successfully exploiting module is related to several specific vulnerabilities (CVE IDs) shown in Table 1. These vulnerabilities are related to corresponding rules in Snort.

Rule Selection: We investigate different rule selection algorithms to pick the one that best fits our assumption. In particular, we consider two alternatives for such rule selection.

Rationale of the two proposed rule selection methods: We believe that the two presented rule selection methods in this work captures the real scenarios of penetration testing, i.e., whether the analyst shows the most dangerous vulnerabilities to test or randomly select rules with no preference.

The most intuitive way for rule selection is that we treat each rule fairly and choose one of them randomly. The Random Selection Algorithm for such process is shown below. Such selection can help when there is no more information available to the security analyst.

Besides random selection, we also try to rank each rule according to the number of its relative vulnerabilities and greedily add these

Table 1: Vulnerability Analysis

CVE ID	CVSS v2.0 Base Score	Affected Software
CVE-2019-15262	7.8	Cisco Wireless LAN Controller
CVE-2015-0936	7.5	Acrobat Reader
CVE-2016-1561	5	ExaGrid EX10000E
CVE-2016-1560	10	ExaGrid EX10000E
CVE-2012-1493	7.8	F5 Big-ip Global Traffic Manager
CVE-2019-17666	8.3	NetApp Active IQ PAS
CVE-2019-17436	6.6	Red Hat OpenStack Platform
CVE-2017-9462	9	Mercurial
CVE-2012-3579	7.9	Messaging Gateway
CVE-2016-7456	10	Vsphere Data Protection
CVE-2012-5975	9.3	Tectia Server
CVE-2006-2407	7.5	wodSSHServer
CVE-2006-2408	7.5	Raydium

Algorithm 1: Random Rule Selection Algorithm

Input: Rule list relative to exploited vulnerabilities
while *Rule list is not empty* **do**
  1. Randomly select a rule from remaining rules in rule list
  2. Add selected rule to configuration of IDS
  3. Remove selected rule from remaining rule list
Output: Rule list added in configuration of IDS

rules from the one that can detect most vulnerable exploiting sessions to the one that can detect least. Such selection should be ideally better than the random selection method. Thus, it is expected to be leveraged by more security experts. If there is a draw during the greedy selection process, we prioritize the rule to solve the vulnerability with higher CVSS score. The Pseudo code for the Greedy Selection Algorithm is shown below.

Algorithm 2: Greedy Rule Selection Algorithm

Input: Rule list relative to exploited vulnerabilities
while *Remaining rule list is not empty* **do**
  Maximum counted number = 0
  for *Rule in remaining rule list* **do**
    Count the number of this rule's relative vulnerabilities
    if *Current counted number > Maximum counted number* **then**
      Selected rule = current rule
      Maximum counted number = current counted number
  Add selected rule to configuration of IDS
  Remove selected rule from remaining rule list
Output: Rule list added in configuration of IDS

Estimation of probability of successful attack: Every time we add a new rule in the configuration of IDS, we will use the db_autopwn plugin to automatically exploit the victim machine and record the number of remaining successful module N. As a result, we can get the probability of successfully attack, which is defined by,

$$p_{a,d} = \frac{N}{N_0} \tag{2}$$

For each probability of successfully attack we measure the maximum CPU utilization caused by IDS of victim machine. This illustrates the security investment for our penetration testing.

Generalization using different Intrusion detection: To illustrate the generalization of our findings, we use Suricata to replace Snort to do all the previous steps again.

4 RESULT

In this section, we present our results which verify that the probability of successful attack decreases exponentially with the investment made by defenders. We also show that the concept of sensitivity do exist in our experiments. For obtaining the results in this section, we use two different IDS to implement same experiment (as explained earlier). The similar results show the generalization of our findings.

4.1 Snort

We use Snort in experiments first. Based on Figure 6 and Figure 7, random selection results in linear relationship between the probability of successful attack and the investment made by defenders while greedy selection results in exponential relationship. Therefore, Greedy selection fits the probability of attack function assumption better. Note that Figure 7 shows that probabilities of successful attack for both targets decrease exponentially with maximum CPU utilization caused by Snort. Meanwhile, different slopes illustrate the sensitivity assumption in which each virtual machine has a different condition.

To represent the relation between variables using specific formula, we use exponential regression to fit the equation to our data, which is the process of finding the equation of the exponential function that fits best for a set of data in (1).

Here are the two equations that we got from the regression for the two different victim machines.

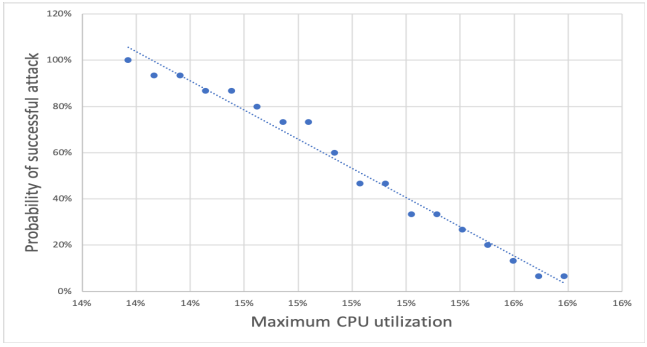

Figure 6: Penetration testing results for random selection using Snort.

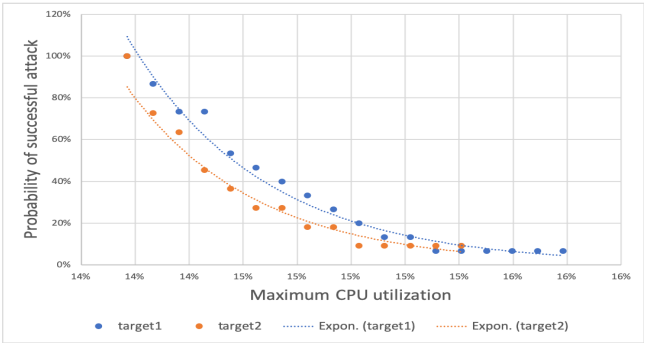

Figure 7: Penetration testing results for greedy selection at different targets using Snort.

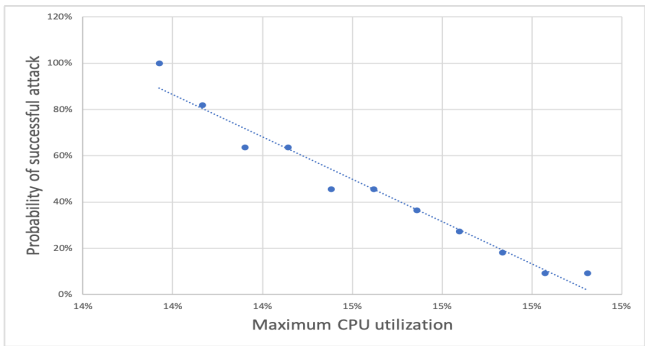

Figure 8: Penetration testing results for random selection using Suricata.

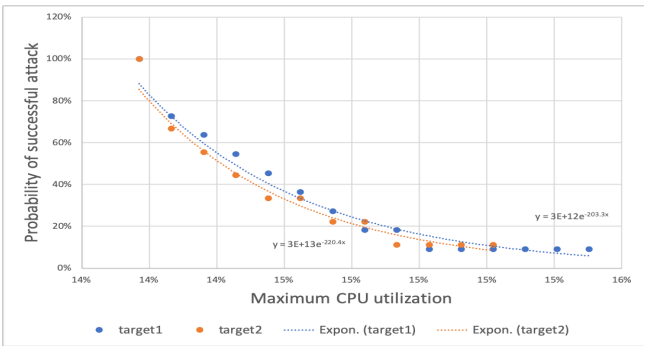

Figure 9: Penetration testing results for greedy selection at different targets using Suricata.

$$p_1(x) = 2 \times 10^{12} \times exp(-198.6 \times x). \quad (3)$$

$$p_2(x) = 8 \times 10^{12} \times exp(-210.9 \times x). \quad (4)$$

In our experiments, there are 17 rules relative to the vulnerability we discovered for victim machines. Therefore, the number of rules applied in Snort is in range of $[0, 17]$. According to relationship shown in Figure 4, the range of maximum CPU utilization caused by Snort is $[14\%, 16\%]$.

4.2 Suricata

Suricata also shows the similar results as Snort. Figure 8 and Figure 9 support that greedy selection fits our assumption better. Again, Figure 9 shows that probabilities of successful attack for both targets decrease exponentially with maximum CPU utilization as well. There are different slopes which illustrate the sensitivity assumption.

The two regression equations we got using the data of Suricata for the two different victim machine are as follows.

$$p_1(x) = 3 \times 10^{12} \times exp(-203.3 \times x). \quad (5)$$

$$p_2(x) = 3 \times 10^{13} \times exp(-220.4 \times x). \quad (6)$$

There are 11 rules relative to the vulnerability in the experiments using Suricata. The range of maximum CPU utilization is $[14\%, 16\%]$.

In sum, such usage of both Snort and Suricata validated the hypothesis that probabilities of successful attack for both targets decrease exponentially with the security investments.

5 DISCUSSION

5.1 Vulnerabilities in CPS and Insider Threats

In CPS such as smart grid system [13, 16], each piece of equipment is accompanied by a Human Machine Interface (HMI), the only gateway through which the equipment can be controlled. The failure scenario in such CPS is triggered when the attacker gets access to the HMI. The vulnerability of the system may arise due to various reasons, such as hacking of the HMI, or an insider attack. Once the attacker gets access to the system, she changes the system settings and gets physical access to the system equipment so that they continue to provide power even during a power system fault. Through this manipulation, the attacker can cause physical damage to the system. Thus, we believe that performing penetration testing can help our understanding of successful attacks on these CPS and

how to eliminate the related threats, including insider attackers or external-based threats.

5.2 Penetration Testing and Preventing Insider Threats

Insider threats range from malicious actors stealing information and compromising security systems to accidental data loss due to employee actions. These insider threats, in most cases, can be prevented or mitigated with effective penetration testing. In particular, frequent penetration testing can prevent insider threats. These tests can check for potential vulnerabilities within the system and eliminate any security threats. They can quickly identify and expose assets that have the potential to be compromised. After these tests have been performed, the security decision-makers can mitigate any loopholes using security software.

6 CONCLUSION

Security of CPS is an important task. In this work, we performed real penetration testing experiments to verify the shape of the relationship between the probability of successful attack on targets and the security investment made by defenders. We considered two intrusion detection systems, Snort and Suricata and two rule selection algorithms (random and greedy). Our experimetns have shown that the probability of successful attack decreases exponentially with the investment made by defenders. Our experiments also have shown that the concept of sensitivity do exist for different virtual machines in our experiments. Future avenue of research includes incorporating other rule selection methods (e.g., testing what happens if we select rules from the lowest CVSS score instead of the greedy approach and presorting rules before selection step). Also, incorporating human effect in real applications to explore prioritizing the vulnerabilities in our experiments would be another avenue for future research.

7 ACKNOWLEDGMENT

This material is based in part upon work supported by the National Science Foundation under Grant Number CNS-1718637. Xingchen Wang was supported under Research Experience for Undergraduates (REU) supplementary fund while performing this research at Purdue University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] M. Abdallah, T. Cason, S. Bagchi, and S. Sundaram. The effect of behavioral probability weighting in a sequential defender-attacker game. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3255–3260, 2020.

[2] M. Abdallah, T. Cason, S. Bagchi, and S. Sundaram. The effect of behavioral probability weighting in a simultaneous multi-target attacker-defender game. In *2021 European Control Conference (ECC)*, pages 933–938. IEEE, 2021.

[3] M. Abdallah, W. J. Lee, N. Raghunathan, C. Mousoulis, J. W. Sutherland, and S. Bagchi. Anomaly detection through transfer learning in agriculture and manufacturing iot systems. *arXiv preprint arXiv:2102.05814*, 2021.

[4] M. Abdallah, P. Naghizadeh, A. R. Hota, T. Cason, S. Bagchi, and S. Sundaram. The impacts of behavioral probability weighting on security investments in interdependent systems. In *2019 American Control Conference (ACC)*, pages 5260–5265. IEEE, 2019.

[5] M. Abdallah, D. Woods, P. Naghizadeh, I. Khalil, T. Cason, S. Sundaram, and S. Bagchi. Morshed: Guiding behavioral decision-makers towards better security investment in interdependent systems. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, page 378–392, New York, NY, USA, 2021. Association for Computing Machinery.

[6] M. Abdallah, D. Woods, P. Naghizadeh, I. Khalil, T. Cason, S. Sundaram, and S. Bagchi. Tasharok: Using mechanism design for enhancing security resource allocation in interdependent systems. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 249–266, 2022.

[7] L. Allen, T. Heriyanto, and S. Ali. *Kali Linux–Assuring security by penetration testing*. Packt Publishing Ltd, 2014.

[8] B. Arkin, S. Stender, and G. McGraw. Software penetration testing. *IEEE Security & Privacy*, 3(1):84–87, 2005.

[9] Y. Baryshnikov. IT security investment and Gordon-Loeb's 1/e rule. In *Workshop on Economics and Information Security (WEIS)*, 2012.

[10] D. Day and B. Burns. A performance analysis of snort and suricata network intrusion detection and prevention engines. In *Fifth International Conference on Digital Society, Gosier, Guadeloupe*, pages 187–192, 2011.

[11] L. A. Gordon and M. P. Loeb. The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):438–457, 2002.

[12] N. Jaswal. *Mastering Metasploit*. Packt Publishing Ltd, 2014.

[13] S. Jauhar, B. Chen, W. G. Temple, X. Dong, Z. Kalbarczyk, W. H. Sanders, and D. M. Nicol. Model-based cybersecurity assessment with nescor smart grid failure scenarios. In *Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*, pages 319–324. IEEE, 2015.

[14] D. Kennedy, J. O'gorman, D. Kearns, and M. Aharoni. *Metasploit: the penetration tester's guide*. No Starch Press, 2011.

[15] J. Kim, H. Kim, K. Lakshmanan, and R. Rajkumar. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. In *Proceedings of the ACM/IEEE 4th international conference on cyber-physical systems*, pages 31–40, 2013.

[16] T. Krause, R. Ernst, B. Klaer, I. Hacker, and M. Henze. Cybersecurity in power grids: Challenges and opportunities. *Sensors*, 21(18):6225, 2021.

[17] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[18] V. Melas. Optimal designs for exponential regression. *Statistics: A Journal of Theoretical and Applied Statistics*, 9(1):45–59, 1978.

[19] H. Moore. Metasploitable 2 exploitability guide. *Retrieved June*, 27:2013, 2012.

[20] A. Quarto, D. Soldo, S. Gemmano, R. Dario, V. Di Lecce, C. Guaragnella, A. Cardellicchio, and A. Lombardi. Iot and cps applications based on wearable devices. a case study: Monitoring of elderly and infirm patients. In *2017 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS)*, pages 1–6. IEEE, 2017.

[21] S. Rahalkar. *Metasploit for Beginners*. Packt Publishing Ltd, 2017.

[22] C. Ramakrishnan and R. Sekar. Model-based vulnerability analysis of computer systems. In *Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation*, volume 128. Citeseer, 1998.

[23] M. Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[24] A. Sanjab, W. Saad, and T. Başar. Prospect theory for enhanced cyber-physical security of drone delivery systems: A network interdiction game. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[25] K. Scarfone and P. Mell. An analysis of cvss version 2 vulnerability scoring. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 516–525. IEEE, 2009.

[26] S. Sinha. Setting up a penetration testing and network security lab. In *Beginning Ethical Hacking with Kali Linux*, pages 19–40. Springer, 2018.

[27] Srinivas. Setting up a pentest Lab with pfSense in virtualBox. https://resources.infosecinstitute.com/topic/setting-pentest-lab-pfsense-virtualbox/, 2015. [Online; accessed 21-October-2021].

[28] D. Stiawan, A. H. Abdullah, and M. Y. Idris. Threat and vulnerability penetration testing: Linux. , 15(3):333–342, 2014.

[29] D. Woods, M. Abdallah, S. Bagchi, S. Sundaram, and T. Cason. Network defense and behavioral biases: an experimental study. *Experimental Economics*, 25(1):254–286, 2022.