American Journal of Software Engineering and Applications
2024, Vol. 12, No. 1, pp. 23-35
https://doi.org/10.11648/j.ajsea.20241201.14

Research Article

Predicting Attack Paths from Application Security Vulnerabilities Using a Multi-Layer Perceptron

Santanam Kasturi[1, *], Xiaolong Li[2], Peng Li[3], John Pickard[3]

[1]Department of Technology Management, Indiana State University, Terre Haute, USA
[2]Department of Electronics and Computer Engineering, Indiana State University, Terre Haute, USA
[3]Department of Technology Systems, East Carolina University, Greenville, USA

Abstract

This paper is in the series of continuing research and proposes an approach to predicting possible attack paths from application security vulnerability-based attack trees. The attack trees are formed by stringing together weaknesses discovered in an application code and a group of applications within a domain. The Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) linked together as a string of vulnerabilities in the attack trees can be visualized as pathways for attacks. These pathways become potential attacks that can spread vertically and horizontally leading to a multi-path attack that can involve multiple software applications. With more data, and huge number of vulnerabilities, it will become impossible to identify all attack paths unless a full-scale implementation of an autonomous processing mechanism is in place. Machine Learning (ML) and Deep Learning (DL) techniques have been adopted in the cybersecurity space for decades, however all the studies have been around networks, endpoints, and device monitoring. This paper focuses on application security and building on earlier work cited, the use of a vulnerability map that uses attack vectors in a Deep Learning (DL) method implementing a Multi-Layer Perceptron (MLP) forms the basis for developing a predictive model that relates a set of linked vulnerabilities to an attack path. The results are encouraging, and this approach will help in identifying successful or failed attack paths involving multiple applications, isolated or grouped, and will help focus on the right applications and the vulnerabilities associated as priority for remediation.

Keywords

Attack Surface, Attack Path, Attack Vector, Vulnerability Map, Deep Learning, Artificial Neural Network, Multi-Layer Perceptron

1. Introduction

There exists an exhaustive amount of work in cybersecurity, and more importantly application security, of various deep learning methods to identify patterns and signatures of attacks. Some relevant literature has been cited for an initial summarization of such methods focusing on multiple problems in cybersecurity that includes discouraging the notion of treating an attack as an isolated incident, of detecting malware behavior, of resolving performance issues with recurrent neural

*Corresponding author: skasturi@sycamores.indstate.edu (Santanam Kasturi)

Received: 5 February 2024; **Accepted:** 30 April 2024; **Published:** 30 May 2024

networks used for analytics, of looking at system behavior to detect anomaly in real time, and many more. The overwhelming observation is an emphasis on using DL methods that reliably provide us promising results for evolving an autonomous solution for attack prediction. Some citations are discussed starting with a comprehensive survey on deep learning methods for cybersecurity.

Starting with a survey paper [1], and associated research [1, 4], that discusses different types of cyber applications, attacks and outcomes using DL methods in detail including detection and classifications [2]. Malware, botnets, network intrusions, drive by download attacks, file type detections, verification if keystrokes were human entry, network traffic identifications, SPAM identification, insider threat detection, user authentication, border gateway protocol anomaly detection, false data injection attack detection [1]. Associating unusual system behavior due to high consumption of resources shown by way of a CPU utilization, or unexplained volume, performance degradation, growing server heap size can all be because of malware injection. A neural network model for a mechanism of visual pattern recognition is proposed in the previous study [3]. The network is self-organized by "learning without a teacher" and acquires an ability to recognize stimulus patterns based on the geometrical similarity of their shapes without being affected by their positions. The Artificial Neural Network (ANN) can be trained to recognize these patterns by way of an optical character recognition-based algorithm.

The main inadequacy current approaches are that they treat the different attack types in isolation [1]. The proposed approach in this research extends the concept of cascading connection of malicious activities throughout an attack lifecycle (e.g., breach, exploitation, command and control, data theft, etc.,) through a vulnerability map that links security vulnerabilities in multiple software applications as an attack tree [5-7]. This approach can be used to predict attack paths using a Multi-Layer Perceptron (MLP) implementation. Two scenarios are cited to highlight the challenges in identifying a dormant attack, which unpacks the payload after staying inactive for some time, botnets, and malware. The best solution for this is to follow the attack path as defined by vulnerability linkages, and then predict the most likely path that an attacker could use to exploit weakness in the application code [6, 7]. Once an initial exploitation has been achieved, the attacker can then use other vulnerabilities to travel laterally. Depending upon other vulnerabilities found along the path, the attacker can begin unpacking the payload that can be a malicious code injection / SQL injection or crippling a system by overwhelming the resources causing a denial of service (DoS). A Cross-site scripting (XSS) attack is another most commonly employed attack used by threat actors. Correlation studies have revealed there exists a pattern for incoming transactions that are blocked by a WAF as a malicious request with a target host application that has a vulnerability [5]. If a WAF blocks a malicious SQLI request and the attack path indicates that the target is a hosted application that has a SQLI vulnerability, the attacker would then try another path. This path could be from another weakness that can be exploited and then travel sideways to the same weakness that was protected by a blocked request by the WAF in the first scenario. That attacker has succeeded in reaching the target. If there is prior knowledge of all likely attack paths, then it can be predicted. All predicted attack paths can be addressed by remediating vulnerabilities, an approach this research is proposing.

Botnets, malware, and system malfunction due to cyber-attacks are all behavioral anomalies and are hard to detect. To detect and identify these anomalies it requires observing the transaction paths, and there can be hundreds of attack paths a threat actor can choose, and it will be buried in a huge volume of vulnerability data. To identify the right attack paths and follow each one of them is a huge task; secondly there are difficulties in identifying what is an anomaly. The reason for the difficulties of these anomalies being detected is discussed briefly.

1. Botnets are a serious problem, and some of them can be dormant for many weeks and go unnoticed before they trigger, after gathering enough data that triggers an action when the critical threshold is reached. As explained above, if an attack pattern can be graphed, then the behavioral analysis approach aims to look at the common patterns that Botnets follow across their life cycle. [8]. This approach can be generalized to detect unseen Botnet traffic. The authors provide an analysis of the viability of Recurrent Neural Networks (RNN) to detect the behavior of network traffic by modeling it as a sequence of states that change over time [8]. Detection of botnets is critical in today's world, especially with the global connectivity, and accessibility of the latest tools of attack available to a threat actor. An undetected Botnet, lying dormant and gathering data quietly until triggered by a global network of neurons, can be catastrophic. The paper also provides a means of using RNN to study the behavior of network traffic to detect unseen Botnets, rather than trying to probe the behavior of a Botnet that can be nearly impossible. So, it is an indirect method of catching the botnet [8].
2. Malware acts in a similar manner to Botnets; it can camouflage and go undetected. Variants can do the trick to avoid anti-virus detection. Past approaches for generating signatures for malware programs were reviewed, and a novel method based on deep belief networks is proposed [2]. Instead of targeting malware, the authors have proposed a deep learning algorithm that detects behavior of malware rather than the signature or token-based approaches for detecting the malware [2].

Considering the above cited scenarios involving botnets and malware, it becomes imperative to identify all exploitable vulnerabilities and prioritize their remediation immediately. So, based on earlier cited works, developing an attack tree to mimic a threat model [5-7, 9, 10], adopting DL methods to predict an attack [11, 12], and using these predictive analytics approaches to ingest threat alerts into a threat intelligence system [6, 7, 13, 14] are critical for improving the security posture of the organization. The proposed research work aims to develop that concept into a tangible approach suitable for application security. Future research opportunities related to developing new datasets to motivate work in developing new DL approaches for cyber security are identified. The need for approaches to be developed that take the adversary into consideration as to how they may use DL as a tool to subvert DL detection mechanisms becomes necessary [1].

2. Predicting Attack Path from the Neural Network Analysis

With more data, and huge number of vulnerabilities, it will become impossible to identify attack paths that a threat actor would exploit unless a full-scale implementation of an autonomous processing mechanism is implemented. Machine Learning (ML) and Deep Learning (DL) techniques have been adopted in the cybersecurity space for decades, however as mentioned before all the studies have been in the area of networks, endpoints, and device monitoring. As far as application security is concerned, DL and Artificial Intelligence (AI) studies are limited to detection and monitoring techniques. This is at the very front end and does not solve what happens deep down in the code and the hosting environment where the code interacts with the systems. Pursuant to the methods described in the previous sections where a data platform consolidating all application security vulnerabilities and further to that doing a correlation analysis, a missing piece in the end analysis is the predictive modeling and feeding the analysis into a threat intelligence and incident response system [13].

2.1. Attack Path Architecture

Determining attack paths from attack trees is a gradual step by step method using a neural network. A multi-layer perceptron is used here to analyze vulnerability data and attack trees to identify and predict the possible attack paths. The first step is to identify the attack vectors in an application security context that allow threat actors to attack an organization for whatever motive, and it is necessary to identify those [15]. The pathway for a threat actor is to exploit open vulnerabilities that exist in any exposed attack surface. In a web application the attack surface is a vast space, and reconnaissance by a threat actor would reveal a huge number of software security issues. Now, Cyber-attacks are no longer a random recon activity or short time attack that exposes lack of controls. These are now called as Advanced Persistent Threats (APT) where an attacker enters a system through a weakness that is easily exploitable, stays dormant till a more possible avenue or pathway to spread laterally provides an opportunity to make a severe attack. While literature covers more of end point attack vectors, the motive and modus operandi of attacks through attack vectors is common to all cyber-attacks [16]. When dealing with a complex prediction problem requiring higher level feature extraction, DL networks are used [9]. As an example, the problem of detecting neurons lying dormant and embedded in a commonly used business application and disguising as human users, has strong implications. Neurons can collect data on user interactions and use that data for creating a perception that a software is used in a certain way, rather than following the real user's navigation and creating useless business requirements. DL algorithms can help greatly in detecting these neurons and identify APT attacks.

The point to focus on is that vulnerabilities that remain open, even if of lower priority from a remediation standpoint, are still possible pathways for an attack, equally applicable in application security as in infrastructure security [15, 16, 1]. Despite an organization's best efforts to plug in gaps in controls, there will still be some loose ends that a threat actor can see as remaining for too long. Many probabilistic methods exist that model the complexity of an organization and look for solutions, however in a dynamically changing scenario where the network and infrastructure keep adding more pieces, the inter-dependence between the segments of the network and infrastructure become a complex issue to define through any model. The most challenging scenario being a compromised employee in the form of an internal threat, third party and open-source software, and mobile devices that come in thousands [17]. Moreover, APT attacks are designed to be not easily detectible, and operate in stealth where an attacker gains control through initial intrusion, then spread laterally, discover more weakness as they spread to gain control of a wider set of resources, and gain full control through privilege escalation, and finally deploy the payload [18]. In order to visualize the path that possibly will be taken by an attacker it requires an attack path and simulation modeling exercises to predict an attack. Attack trees, threat modeling, and vulnerability visualization maps become essential to understand the full lay of the land [6, 7, 19, 20].

In their paper the authors reinforce the limitations of Web Application Firewall (WAF) and Runtime Application Self Protect (RASP) if they are only dependent on signature, anomaly, policy, and hybrid rules that have advantages and disadvantages [14]. Unless the threat intelligence system is fed with data that projects a possible path to an attack, as proposed in [6, 7, 13], it will still have the same issues in the proposed approach of [14]. A simple reason for that is when dealing with thousands of vulnerabilities that have remained unremedied for many years, and to determine what is the right path for a threat actor will be daunting.

The data flow into the MLP is structured in three steps and is shown in Figure 1. The first step is data collection and classification of vulnerabilities into independent variables and covariates. The second step is developing the attack tree based on threat modeling principles, using CWE-CVE linkages, and application-to-application CWE-CVE linkages [6, 7]. The final step is to define the output variable and ingest the different input variables from the attack tree into the MLP.

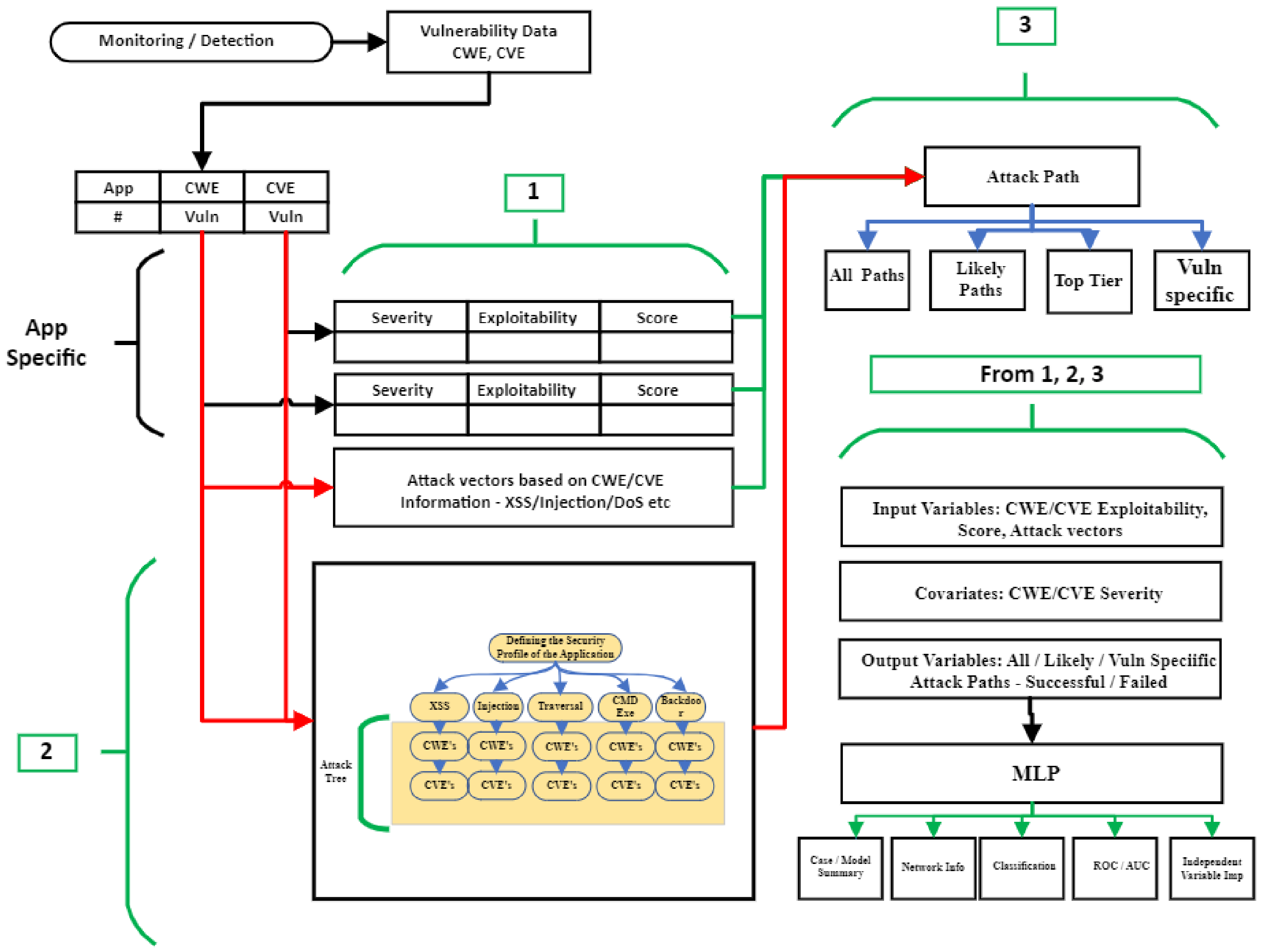
Monitoring / Detection
Vulnerability Data CWE, CVE
App
CWE
CVE
#
Vuln
Vuln
App Specific
1
Severity
Exploitability
Score
Severity
Exploitability
Score
Attack vectors based on CWE/CVE Information - XSS/Injection/DoS etc
2
Defining the Security Profile of the Application
XSS
Injection
Traversal
CMD Exe
Backdoor
CWE's
CVE's
Attack Tree
3
Attack Path
All Paths
Likely Paths
Top Tier
Vuln specific
From 1, 2, 3
Input Variables: CWE/CVE Exploitability, Score, Attack vectors
Covariates: CWE/CVE Severity
Output Variables: All / Likely / Vuln Speciific Attack Paths - Successful / Failed
MLP
Case / Model Summary
Network Info
Classification
ROC / AUC
Independent Variable Imp

Figure 1. An architecture for Attack Path identification incorporating an MLP.

2.1.1. Step 1

In the context of application security, the research proposes classifying the attack vectors into three categories, and associating each vulnerability class to the application where it was discovered, is the first step, Figure 1:

1. The first attack vector, like XSS, SQLI/Code/Command Injection, or DoS vulnerability, is linked to the CWE discovered from detection / testing tools and the associated vulnerability, severity, exploitability, and a CWE score.
2. The second attack vector, like XSS, SQLI/Code/Command Injection, or DoS vulnerability, is linked to the CVE discovered from detection / testing tools and the associated vulnerability.
3. The third attack vector is predominantly an alternate vulnerability besides what is listed in the first attack vector. Some CWEs suggest multiple vulnerabilities, e.g., a CWE could have multiple exploits available to the threat actor, the attack could exploit a weakness to do a code injection that can result in a command execution or a DoS.

2.1.2. Step 2

The attack vectors, the CWE/CVE information, the linkages of the CWE and CVE associated with the application, severity and exploitability of the vulnerabilities associated with the specific CWEs and CVEs in the application, and CWE/CVE score are the variables ingested into developing the attack tree, depending on the security profile of the application, Figure 1, [6, 7, 9, 10, 21-27].

1. The attack tree shows a vulnerability topology, and all the paths that are open to an attacker.
2. Each path uses a CWE-CVE, or a CVE-CVE vulnerability linkages. Applications use multiple open-source components that could have many associated CVEs and could have multiple CWEs discovered through a scan or a test [5].

3. The vulnerability map provides a horizontal and vertical distribution of weakness across many applications, a map that an attacker can build with multiple reconnaissance of the enterprise's exposed attack surface [28].

2.1.3. Step 3

The variables from Step 1 and Step 2 define the attack path, Figure 1. The attack paths are then categorized as all paths, likely paths, and vulnerability specific paths.

1. All attack paths show all the vulnerability linkages from the UI to deep down into the code. This is the parent tree or a forest [6, 7].
2. Likely attack paths are the ones that cross a threshold. The threshold is derived with a combination of factors that suggest a higher risk potential of an easier path for an attacker. Vulnerability scores, severity, exploitability with more than one attack vector in the path.
3. Top Tier Attack paths are a subset of Likely Attack path category with an additional threshold of having CWEs in the KEV Top 10 [27].
4. Vulnerability specific attack paths are an even narrower option for an attacker, but more easily exploitable, as it directly corresponds to a specific vulnerability as a single attack vector that is common for a CWE and CVE. It can be an XSS, SQLI or a DoS weakness that an application can have at the code and component level, meaning the CWE and the CVE could be pointing to a XSS vulnerability [5].

2.2. Keys to Chain Linking CWEs and CVEs

Common Weakness Enumeration (CWE) is a community-developed list of common software (and hardware) weakness where a "weakness" is due to a developer introduced flaw that can be exploited by a threat actor to expose it as a vulnerability. The 2023 CWE Top 25 Most Dangerous Software Weaknesses list points to specific parts of code that are flawed in design as well as in coding and require immediate attention. The "On the Cusp" weaknesses beyond the 25 add up to 40, an additional 15, that in due course could rise to the top 25. The Cybersecurity and Infrastructure Security Agency (CISA) published the "Known Exploited Vulnerabilities (KEV) Catalog in 2021" [27]. These categories of CWEs have been reported through the Common Vulnerabilities and Exposures (CVE) program and are either monitored to be likely or have been exploited and it is recommended that the KEV Top 10 CWEs are continuously monitored for updates along with CWE Top 25 and on-the-cusp lists [21-27].

Using the information provided in MITRE, CISA, OWASP, and NVD data, and the rankings and scores given to each CWE and CVE one can identify what can be a potential high-risk weakness and vulnerability, and address that as a priority [21-27]. When dealing with hundreds of applications having thousands of vulnerabilities chain linking helps in further focusing on attack paths that a threat actor may choose to follow a transaction with intersecting vulnerabilities. It is necessary to identify all attack paths first, and then funnel down to the most likely attack paths. This will help narrow down the list of vulnerabilities and vulnerable paths and ensure these are addressed immediately. In a vulnerability report, a vulnerability is assigned a unique CVE and is categorized into one or many CWEs. Similarly, a CWE could have multiple CVEs referred to it. Grouping vulnerabilities and chain linking them helps fix vulnerabilities on priority.

For example, CVE-2021-37701 is mapped to two CWEs 22, 59. CWE 22 is a path traversal weakness capable of code or command execution from an integrity and availability standpoint and from a confidentiality standpoint it can be either a command / code execution or a denial-of-service weakness. CWE 59 is a link following weakness capable of bypassing protection mechanism from an access control and confidentiality standpoint and command / code execution from an integrity standpoint. As for CVE-2021-37701, it is a directory traversal vulnerability. A likely path for a threat actor to bypass protection is to use CWE 22 as an entry and traverse laterally through CVE-2021-37701 and attack CWE 59. This is called an attack path identified using chain linking of weaknesses and vulnerabilities. In another use case of an attack path, CVE-2023-20863 is mapped to two CWEs 400 and 917. CWE400 is an uncontrolled resource consumption weakness leading to a denial of service or bypass protection mechanism from an availability standpoint. CWE917 is an express language injection weakness that can lead to a command/code injection threat. As for CVE-2023-20863, is again a denial-of-service vulnerability. So, a likely path could be to attack CWE400 to bypass protection, and travel laterally through CVE-2023-20863 causing a denial-of-service attack and further attack CWE917 to inject malicious code. Assigning CWEs to CVEs is another laborious task, especially if there are too many CVEs associated with a CWE. This was done manually in this research but can be performed using an automated solution proposed in [29].

CWE22 is in the top 25 most dangerous weaknesses, CWE400 and CWE59 are in the On-the-cusp list in the top 40. CWE22 is also in the Top 10 KEV list [21-27]. The next step is to define dependent, independent, and covariate variables.

2.3. MLP Implementation in This Research and the Associated Variables

An Artificial Neural Network conceptually resembles the human brain and is designed structurally to function and mimic a brain's identification process and trains itself to perfect the identification process to recognize variants. The knowledge it acquires in the training process is stored as synaptic weights and each neuron passes the pieces of the knowledge to the output and make the whole identification process complete. The multi-layer perceptron (MLP) is one of the reliable implementations of neural networks and has been widely used in medical sciences, weather forecasting, pattern recognition, multi-spectral image processing, experimental physics, and many other fields of science, engineering, and technology [11, 30-32].

IBM SPSS Neural Network is used in this work [33]. The neural network module in Statistical Product and Service Solutions (SPPS) has two features:

1. MLP
2. Radial Basis Function

The MLP has been used in this research [33]. MLP's are a supervised network, and they require an expected output in order to learn and train. The expected outcome is split into training and test components, and once the network has trained, it then uses its training knowledge and applies it to test and predict for the remainder of the cases. MLPs consist of an input layer with neurons (independent input variables), an output layer with neurons (dependent variables), and one or more hidden layers containing neurons to help capture the nonlinearity in the data. The discussion that follows explains the step-by-step approach to include various parameters to see how the neural network's prediction improves or degrades. Four runs of the model were executed to study how the data behaved and to analyze the prediction accuracy. Optimal execution is determined as an intermediate assessment between when the model underperforms or over performs where the predictions tend to be less accurate than when the model performs optimally. There are many types of activation functions that can be used for the hidden layers, however non-linear activation functions differentiate the complex relationship that may exist in the variables. The executions are termed as 'runs' and are labeled as such. With each run with the same variables, the model uses the model parameters and training data to retrain for better prediction. For this implementation, Run 1 - Run 3 the hidden layer function that was used is the Hyperbolic Tangent function, while Run 4 was executed using the Sigmoid activation function for the hidden layer. While both activation functions produced similar results, the Hyperbolic Tangent function is recommended as better performing and for faster convergence of the learning algorithm in an MLP [31, 32].

2.3.1. Variables and Runs

Section 2.1, 2.2, and 2.3 provided all the needed information to define the variables to set up the MLP and do the model "runs." Here is a summary of the variables that will be used in the runs in this research.

1. Dependent Variables: All Attack Paths, Likely Attack Paths, and Top Tier Attack paths (for this research not included in scope is predicting Vuln Specific Attack Paths)
2. Covariates: CWE and CVE severity scores
3. Independent Variables: CWE and CVE scores, CWE and CVE exploitability scores, attack vector 1, attack vector 2, and attack vector 3 (based on vulnerability and weakness classification, defined in section 2.1.1 Step 1.

The details of the individual 'runs' along with the MLP results are given in the following sections.

2.3.2. Run-1

Run 1: With the hidden layer activation function chosen as a Hyperbolic Tangent, dependent variables are the All Attack Paths, Likely Attack Paths, and Top Tier Paths; covariates are Vulnerability Severities (of CWE and CVE). The independent variables are Exploitability (known from CWE and CVE), CWE and CVE scores, KEV Top 10, and Attack Vector 1 and Attack Vector 2 (as explained in section 2.1.1 step 1).

2.3.3. Run-2

Run 2: With the hidden layer activation function chosen as a Hyperbolic Tangent, dependent variables are the All Attack Paths, Likely Attack Paths; and Top Tier Paths, covariates are Vulnerability Severities (of CWE and CVE). The independent variables are Exploitability (known from CWE and CVE), CWE and CVE scores, KEV Top 10 and Attack Vector 1, Attack Vector 2, and Attack Vector 3 (as explained in section 2.1.1 step 1).

2.3.4. Run-3

Run 3: Is a re-run of Run 2 executions with the hidden layer activation function chosen as a Hyperbolic Tangent, where dependent variables are the All Attack Paths, Likely Attack Paths; and Top Tier Paths covariates are Vulnerability Severities (of CWE and CVE). The independent variables are Exploitability (known from CWE and CVE), CWE and CVE scores, KEV Top 10 and Attack Vector 1, Attack Vector 2, and Attack Vector 3 (as explained in section 2.1.1 step 1).

2.3.5. Run-4

Run 4: Is a re-run of Run 3 executions with the hidden layer activation function chosen as a Sigmoid, where dependent variables are the All Attack Paths, Likely Attack Paths; and Top Tier Paths, covariates are Vulnerability Severities (of CWE and CVE). The independent variables are Exploitability (known from CWE and CVE), CWE and CVE scores, and Attack Vector 1, Attack Vector 2, and Attack Vector 3 (as explained in section 2.1.1 step 1).

3. Summary of Results of the MLP Implementation

The objective of using an MLP is to predict the attack paths. When the CWE-CVE relationship is clearly established it is easy to create an attack path by linking the vulnerabilities. The vulnerabilities defined in CVE are mapped to a CWE that describes a weakness in the code that is sometimes not very intuitive and needs to be tracked with Common Attack Pattern Enumeration and Classification (CAPEC) that stores attack patterns. These attack patterns provide details of how an attack can take shape through the CWE weakness. CAPEC attack patterns are descriptions of common attributes and approaches employed by adversaries to exploit known weaknesses. The challenge however is that the CVE vulnerability can be linked to the attack pattern only through a CWE for extracting CAPEC attack patterns. This was done manually, however an approach to link the two repositories is proposed by tracing CVE vulnerability information to CAPEC attack patterns using language processing techniques [12]. In the current research this was done manually to establish the linkages.

The details of the individual 'runs' are given in the following section.

1. For all runs, the model adjusted to just one middle layer, and depending upon the number of independent variables the number of nodes were also automatically determined. This is in view of the data size. The sample included vulnerability data for only seven applications. With more applications and a larger volume of data, the MLP would require more hidden layers, with more nodes.
2. Run 2 (Table 1, Table 2, and Table 3) and Run 3 have the same variables and network structure. Run 2 had 74.5 % training and 25.5 % testing samples, Table 1. Run 3 showed confidence in the model by using less data samples for training compared to Run 2, Table 4, and this is due to the fact Run 3 was executed with same structure and variables after Run 2. Run 3 had 70% training and 30 % testing samples and yet performed better. This shows the model had trained well from Run 2 and had more confidence in the training with less samples in Run 3. The overall percentage correct for training in Run 2 was 100% and for testing was 95.7%, Table 3. The overall percentage correct for training in Run 3 was 99.3% and testing was 99.7%, Table 5. The model adjusted on the variables of importance based on the training from Run 2, Table 6, Figure 2 represent data for Run 2; Table 7 and Figure 3 represent Run 3.

Table 1. *Run 2 Case Processing Summary.*

		N	Percent
Sample	Training	120	75.5%
	Testing	39	24.5%
Valid		159	100.0%
Excluded		145	
Total		304	

Table 2. *Run 2 Network Information.*

Input Layer	Factors	1	Attack Vector 1
		2	Attack Vector 2
		3	Attack Vector 3
		4	CWE Scores
		5	CVE Scores
		6	CWE Exploitability Score
		7	CVE Exploitability Score
	Covariates	1	CWE Severity
		2	CVE Severity
	Number of Units[a]		86
	Rescaling Method for Covariates		Standardized
Hidden Layer(s)	Number of Hidden Layers		1
	Number of Units in Hidden Layer 1[a]		3
	Activation Function		Hyperbolic tangent
Output Layer	Dependent Variables	1	All Attack Paths
		2	Likely Attack Paths
		3	Top Tier Paths
	Number of Units		6
	Activation Function		Softmax
	Error Function		Cross-entropy

Table 3. Run 2 Overall Percent Correct.

Sample	Overall Percent Correct
Training	100.00%
Testing	95.70%

Table 4. Run 3 Case Processing Summary.

		N	Percent
Sample	Training	112	70.00%
	Testing	48	30.00%
Valid		160	100.00%
Excluded		144	
Total		304	

Table 5. Run 3 Overall Percent Correct.

Sample	Overall Percent Correct
Training	99.70%
Testing	99.30%

Table 6. Run 2 Independent Variable Importance.

	Importance	Normalized Importance
Attack Vector 1	0.136	76.20%
Attack Vector 2	0.084	47.00%
Attack Vector 3	0.065	36.60%
CWE Scores	0.171	95.30%
CVE Scores	0.133	74.60%
CWE Exploitability Score	0.055	30.80%
CVE Exploitability Score	0.179	100.00%
CWE Severity	0.031	17.20%
CVE Severity	0.145	81.30%

Table 7. Run 3 Independent Variable Importance.

	Importance	Normalized Importance
Attack Vector 1	0.146	77.90%
Attack Vector 2	0.077	41.10%

	Importance	Normalized Importance
Attack Vector 3	0.104	55.30%
CWE Scores	0.188	100.00%
CVE Scores	0.146	77.60%
CWE Exploitability Score	0.115	61.00%
CVE Exploitability Score	0.112	59.70%
CWE Severity	0.017	9.30%
CVE Severity	0.094	50.20%

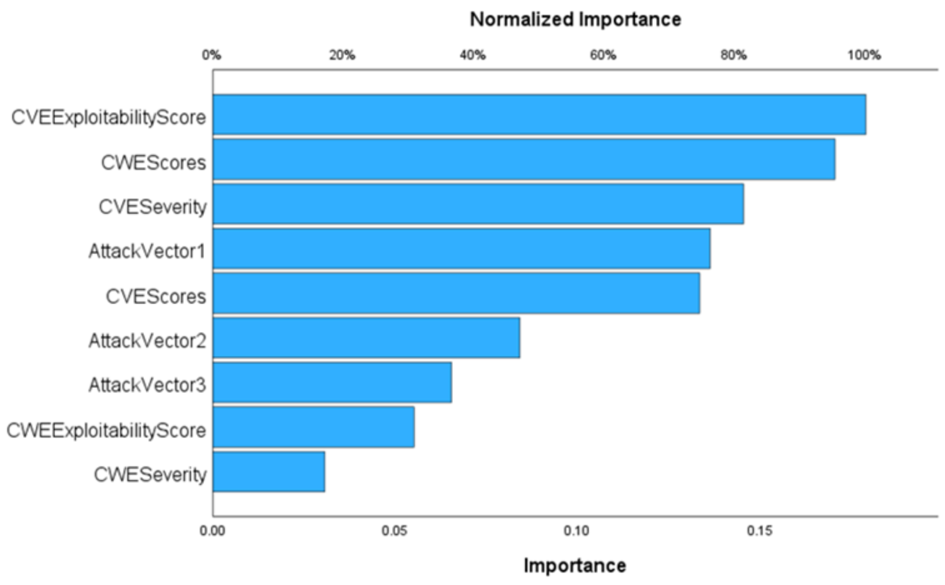

Figure 2. Run 2 Normalized Importance.

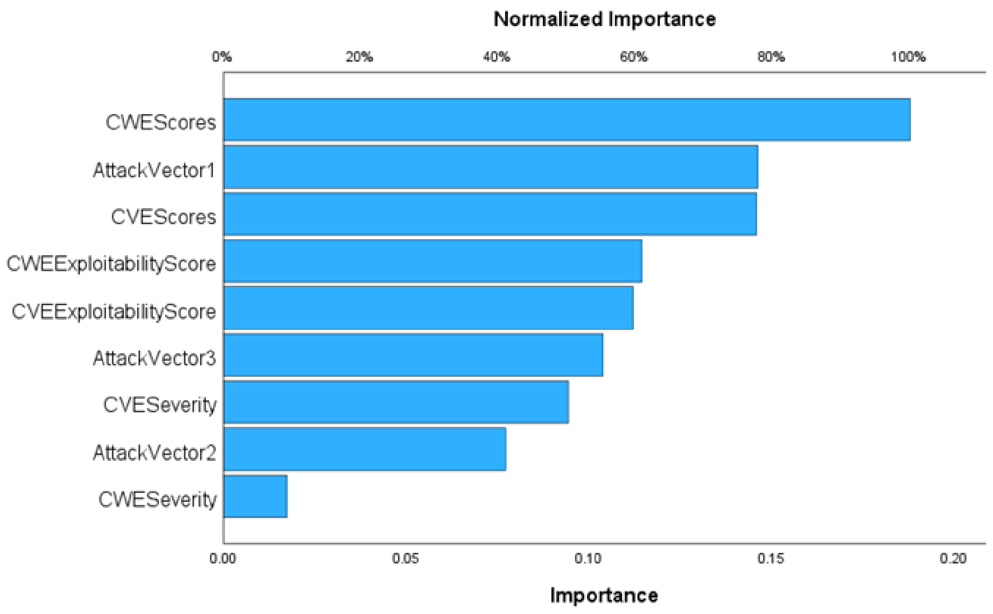

Figure 3. Run 3 Normalized Importance of Independent Variables.

Run 4 was also executed by applying the activation function for the hidden layer as Sigmoid to study the performance. Although no perceivable difference is seen, it is recommended to use the Hyperbolic Tangent function [31, 32]. Compared to Run 3, it is seen the training sample percentage increased to 71.3% and the testing sample percentage dropped to 28.7, Table 8. The model is trying to adapt to the new function by increasing the training samples. The overall percentage correct is 100 % for training and 97% for testing, Table 9, compared to Run 3 which had 99.7 % for training and 99.3 % for testing. The top 5 influencing independent variables for Run 3 are CWE Scores, Attack Vector 1, CVE Scores, CWE Exploitability, and CVE Exploitability as shown in Figure 3. For Run 4 the top 5 influencing independent variables are CWE Scores, CVE Exploitability Scores, CVE Severity, Attack Vector 1, and Attack Vector 2 as seen in Figure 4. The key output is the Receiver Operating Characteristics (ROC) and Area Under the Curve (AUC). The ROC for Likely Paths and Top Tier Paths for Hyperbolic Tangent function and the Sigmoid function for Run 3 and Run 4 showed perfect prediction. The ROC and AUC for the All Attack Paths for both functions showed a slight difference, but nothing significant. For Run 3 the ROC using Hyperbolic Tangent functions for All Attack Paths is shown in Figure 5 and the AUC in Table 11. For Run 4 using the Sigmoid function the All Attack Paths is shown in Figure 6 and the AUC in Table 12. The performance is comparable, and the prediction is excellent. This method can be implemented in larger efforts with more applications.

Table 8. *Run 4 Case Processing Summary.*

		N	Percent
Sample	Training	114	71.30%
	Testing	46	28.70%
Valid		160	100.00%
Excluded		144	
Total		304	

Table 9. *Run 4 Overall Percent Correct.*

Sample	Overall Percent Correct
Training	100.00%
Testing	97.10%

Table 10. *Run 4 Independent Variable Importance.*

	Importance	Normalized Importance
Attack Vector 1	0.107	50.40%
Attack Vector 2	0.106	50.20%
Attack Vector 3	0.064	30.00%
CWE Scores	0.212	100.00%
CVE Scores	0.099	46.70%
CWE Exploitability Score	0.075	35.30%
CVE Exploitability Score	0.166	78.40%
CWE Severity	0.011	5.20%
CVE Severity	0.161	76.20%

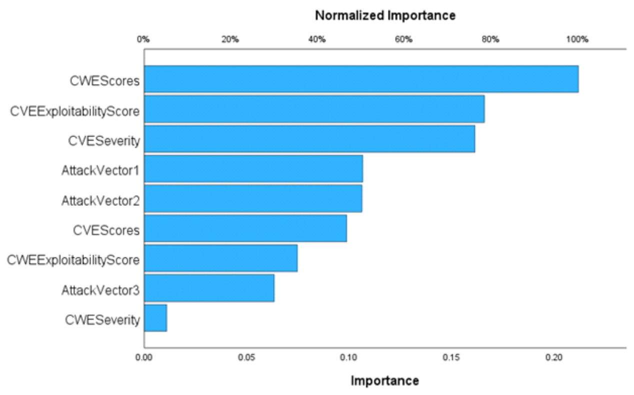

Figure 4. *Run 4 Normalized Importance of Independent Variables.*

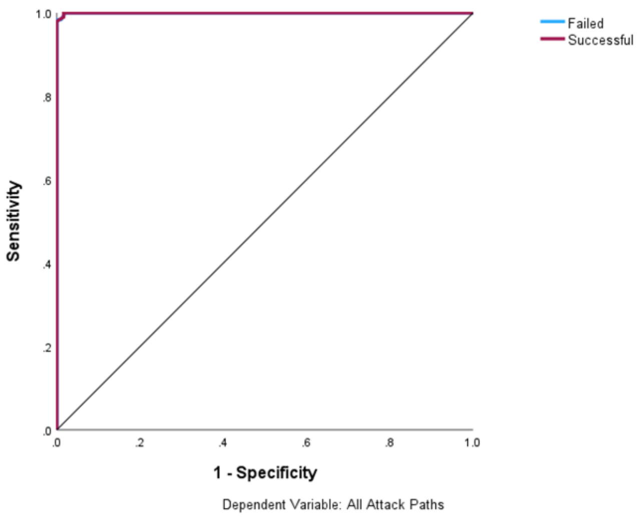

Figure 5. *Run 3 ROC for All Attack Paths.*

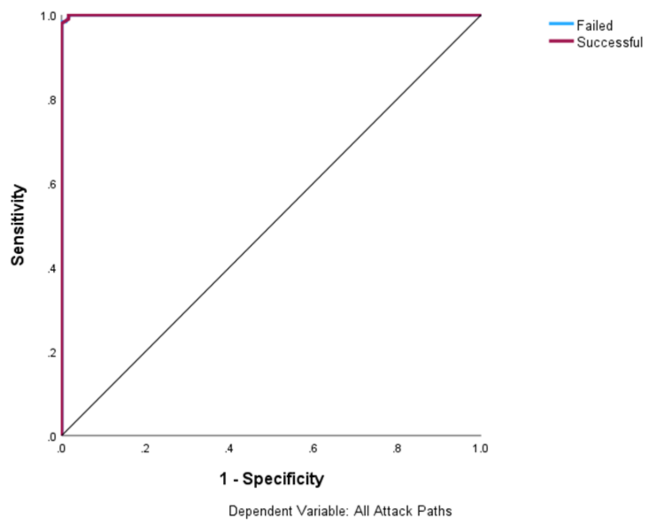

Figure 6. *Run 4 ROC for All Attack Paths.*

Table 11. *Run 3 Area Under the Curve for All Attack Paths.*

		Area
All Attack Paths	Failed	1.000
	Successful	1.000
Likely Attack Paths	Failed	1.000
	Successful	1.000
Top Tier Paths	Failed	1.000
	Successful	1.000

Table 12. *Run 4 Area Under the Curve for All Attack Paths.*

		Area
All Attack Paths	Failed	0.999
	Successful	0.999
Likely Attack Paths	Failed	1.000
	Successful	1.000
Top Tier Paths	Failed	1.000
	Successful	1.000

Overall, the MLP with available data has predicted the attack path very accurately. The objective of identifying the attack paths that a threat actor could use to penetrate is demonstrated through this research. The next step is to predict an attack and integrate this solution into a threat intelligence system to set up alerts for security operations teams to act. The study of correlation between an incoming WAF monitored requests with existing vulnerabilities as proposed in [5] can now be further explored by following the attack paths identified and looking at the transaction behavior [34]. The need for end point behavior study and correlation of host behavior with application vulnerabilities through a specific attack path becomes essential and can be achieved as an extension of the current research. Another key to achieving this end-to-end integration is to consolidate all application security vulnerabilities information obtained through monitoring, scans, and tests [35]. This will help in easier handling of data for correlation, ML, and enabling autonomous processing of all intermediate segments to chain link the vulnerabilities [5-7, 12, 29, 35].

4. Conclusions

The paper provides a solution to develop a predictive system from attack trees using Deep Learning techniques. By employing an Artificial Neural Network like the MLP the proposed approach has shown how to predict existing and potential attack paths in real time.

The benefits of an attack tree are that it gives the security specialists a topology of vulnerability distribution that is otherwise buried and lost in silos, in an ever-growing security technical debt. Taking the next step forward is to use this topology of vulnerability to identify attack paths. With hundreds of applications and thousands of vulnerabilities it is impossible to process the information and plan for a call for action. An autonomous deep learning technique needs to be adopted to pin-point and focus on specific attack paths that are available to a threat actor. In the proposed approach using an MLP implementation results provide confidence in predicting exploitable attack paths that can help in simulation exercises to determine if the vulnerable attack paths do indeed suggest the likely route an attacker might choose. These likely and top priority attack paths then become the focus of security operations to ingest the exploitability factor into a threat intelligence system.

Future efforts should focus along the lines proposed by this research and findings.

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
AUC	Area Under the Curve
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DOS	Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
KEV	Known Exploited Vulnerabilities

ML	Machine Learning
MLP	Multi-Layer Perceptron
OSS	Open-Source Software
OWASP	Open Worldwide Application Security Project
RASP	Runtime Application Self Protect
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SPSS	Statistical Product and Service Solutions
SQLI	SQL Injection
XSS	Cross-Site Scripting
WAF	Web Application Firewall

Acknowledgments

This research has been supported and guided by Dr. Xiaolong Li of Indiana State University, USA, and Dr. John Pickard and Dr. Peng Li of East Carolina University, USA.

Conflicts of Interest

The authors declare no conflicts of interests.

References

[1] Berman, D. S., Buczak, A. L., Chavis, J. S., and Corbett, C. L. (2019), "A Survey of Deep Learning Methods for Cyber Security", *Information 2019, 10(4), 122; Machine Learning for Cyber-Security*, Available from: https://doi.org/10.3390/info10040122

[2] David, O. E. and Netanyahu, N. S. (2015) "DeepSign: Deep learning for automatic malware signature generation and classification", in *2015 International Joint Conference on Neural Networks, IJCNN 2015, Article 7280815 (Proceedings of the International Joint Conference on Neural Networks; Vol. 2015-September). Institute of Electrical and Electronics Engineers Inc.*, Available from: https://doi.org/10.1109/IJCNN.2015.7280815

[3] Fukushima, K. (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position", *Biol Cybernetics 36*, 193-302 (1980).

[4] Pascanu, R., Mikolov, T. and Bengio, Y. (2013). "On the difficulty of training recurrent neural networks", In *Proceedings of the 30th International Conference on Machine Learning*, 28(3): 1310-1318, Available from https://proceedings.mlr.press/v28/pascanu13.html

[5] Kasturi, S., Li, X., Pickard, J., and Li, P. (2023) "Understanding Statistical Correlation of Application Security Vulnerability Data from Detection and Monitoring Tools", *2023 33rd International Telecommunication Networks and Applications Conference*, Melbourne, Australia, 2023, pp. 289-296, https://doi.org/10.1109/ITNAC59571.2023.10368476

[6] Kasturi, S., Li, X., Li, P., Pickard, J. (2024). "A Proposed Approach to Integrate Application Security Vulnerability Data with Incidence Response Systems", *American Journal of Networks and Communications*, *13*(1), 19-29. https://doi.org/10.11648/j.ajnc.20241301.12

[7] Kasturi, S., Li, X., Pickard, J., Li, P. (2024). "Prioritization of Application Security Vulnerability Remediation Using Metrics, Correlation Analysis, and Threat Model", *American Journal of Software Engineering and Applications*, *12*(1), 5-13. https://doi.org/10.11648/j.ajsea.20241201.12

[8] Torres, P.; Catania, C.; Garcia, S.; Garino, C. G. (2016). "An Analysis of Recurrent Neural Networks for Botnet Detection Behavior", *In Proceedings of the 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires, Argentina, 15–17 June 2016; pp. 1–6*, https://doi.org/10.1109/ARGENCON.2016.7585247

[9] Hajrić, A., Smaka, T., Baraković, S., and Husić, J. B. (2020) "Methods, Methodologies, and Tools for Threat Modeling with Case Study", *Telfor Journal*, Vol. 12, No. 1, 2020.

[10] Xiong, W., Legrand, E., Aberg, O., and Lagerstrom, R. (2022). "Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix", *Software and Systems Modeling (2022)* 21: 157–177 https://doi.org/10.1007/s10270-021-00898-7

[11] SÜT, N., and ÇELİK, Y. (2012). "Prediction of mortality in stroke patients using multilayer perceptron neural networks", *In Turkish Journal of Medical Sciences: Vol. 42: No. 5, Article 20*. https://doi.org/10.3906/sag-1105-20

[12] Kanakogi, K., Washizaki, H., Fukazawa, Y., Ogata, S., Okubo, T., Kato, T., Kanuka, H., Hazeyama, A., Yoshioka, N. (2021). "Tracing CVE Vulnerability Information to CAPEC Attack Patterns Using Natural Language Processing Techniques", *Information 2021, 12, 298*. https://doi.org/10.3390/info12080298

[13] Reversing Labs. (2023). "How to Evaluate Threat Intelligence Feeds, eBook-How-to-Evaluate-Threat-Intelligence-Feeds", *Reversing Labs*, Available from: https://www.reversinglabs.com/resources/how-to-evaluate-threat-intelligence-feeds

[14] Sevri, M., & Karacan, H. (2022). "Two Stage Deep Learning Based Stacked Ensemble Model for Web Application Security," In *KSII Transactions on Internet and Information Systems, vol. 16, no. 2, pp. 632-657*, 2022, http://doi.org/10.3837/tiis.2022.02.014

[15] Suskailo, V., Opirskyy, I., and Vasilylyshyn, S. (2020). "Analysis of the attack vectors used by threat actors during the pandemic," *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, Zbarazh, Ukraine, 2020, pp. 261-264, https://doi.org/10.1109/CSIT49958.2020.9321897

[16] Karantzas, G., and Patsakis, C. (2021). "An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent ThreatsWidely Used Attack Vectors", *J. Cybersecur. Priv.* 2021, 1, 387–421. https://doi.org/10.3390/jcp1030021

[17] Tiwari, V. K., and Dwivedi, R. (2016). "Analysis of cyber attack vectors," *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2016, pp. 600-604 https://doi.org/10.1109/CCAA.2016.7813791

[18] Mern, J., Hatch, K., Silva, R., Hickert, C., Sookoor, T., and Kochenderfer, M. J. (2022) "Autonomous Attack Mitigation for Industrial Control Systems," In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Baltimore, MD, USA, 2022, pp. 28-36*, Available from: https://doi.org/10.1109/DSN-W54100.2022.00015

[19] Kalogeraki, E.-M., Papastergiou, S., and Panayiotopoulos, T. (2022). "An Attack Simulation and Evidence Chains Generation Model for Critical Information Infrastructures". *Electronics 2022*, 11, 404. https://doi.org/10.3390/electronics11030404

[20] Lohmann, P., Albuquerque, C., and Machado, R. C. S. (2023). "Systematic Literature Review of Threat Modeling Concepts", In *Researchgate Conference Paper, March 2023* https://doi.org/10.5220/0000168400003405

[21] MITRE. (2023). "CWE Top 25 Most Dangerous Software Weaknesses", *MITRE*, CWE - 2023 CWE Top 25 Most Dangerous Software Weaknesses mitre.org

[22] MITRE. (2023). "2023 On the Cusp"-Other Dangerous Software Weaknesses", *MITRE*, https://cwe.mitre.org/top25/archive/2023/2023_onthecusp_list.html#top25list

[23] OWASP (2021). "OWASP Top 10. *OWASP",* https://owasp.org/Top10/

[24] MITRE. (2018) "Common Vulnerabilities and Exposures (CVE) Numbering Authority (CNA) Rules", *MITRE*, https://cve.mitre.org/cve/cna/CNA_Rules_v2.0.pdf

[25] Mell, P., Scarfone, K., and Romanosky, S. (2007). "A Complete Guide to the Common Vulnerability Scoring System Version 2.0", *National Institute of Standards and Technology (NIST) and Carnegie Mellon University*, https://www.first.org/cvss/v2/cvss-v2-guide.pdf

[26] First. (2019). "Common Vulnerability Scoring System version 3.1Specification Document Revision 1", by *FIRST. Org*, Inc., https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf

[27] CISA. (2021). "Reducing the Significant Risk of Known Exploited Vulnerabilities", *Cybersecurity and Infrastructure Security Agency (CISA),* https://www.cisa.gov/known-exploited-vulnerabilities

[28] Miller, L. (2023). "Attack Surface Management For Dummies", *Palo Alto Networks Special Edition, 2023 by John Wiley & Sons, Inc., Hoboken, New Jersey.*

[29] Liu, P., Ye, W., Duan, H., Li, X., Zhang, S., Yao, C., and Li, Y. (2023)." Graph neural network based approach to automatically assigning common weakness enumeration identifiers for vulnerabilities", In *Cybersecurity 6, 29(2023).* https://doi.org/10.1186/s42400-023-00160-1

[30] Popescu, M-C., Balas, V., & Perescu-Popescu, L., and Mastorakis, N. (2009). "Multilayer perceptron and neural networks", *In WSEAS Transactions on Circuits and Systems, Vol 8, Issue 7, pp. 579-588,* Available from: https://www.researchgate.net/publication/228340819_Multilayer_perceptron_and_neural_networks

[31] Özkan, C., and Erbek, D.S. (2003) "The Comparison of Activation Functions for Multispectral Landsat TM Image Classification", *American Society for Photogrammetry and Remote Sensing (ASPRS),* https://www.asprs.org/wp-content/uploads/pers/2003journal/november/2003_nov_1225-1234.pdf

[32] Karrach, L., and Pivarčiová, E. (2023). "Using Different Types of Artificial Neural Networks to Classify 2D Matrix Codes and Their Rotations - A Comparative Study", *J. Imaging 2023, 9, 188.* https://doi.org/10.3390/jimaging9090188

[33] IBM. (2021). "IBM SPSS Statistics", *IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US,* Available from: https://www.ibm.com/docs/en/SSLVMB_29.0.0/pdf/IBM_SPSS_Neural_Network.pdf

[34] Vartouni, A. M., Teshnehlab, M., and Kashi, S. S. (2019). "Leveraging deep neural networks for anomaly-based web application firewall", *IET Information Security*, https://doi.org/10.1049/iet-ifs.2018.5404

[35] Kasturi, S., Li, X., Li, P., and Pickard, J. (2024). "On the Benefits of Vulnerability Data Consolidation in Application Security", Vol. 19 No. 1(2024): In Proceedings of *The 19th International Conference on Cyber Warfare and Security*, pp. 455-462, https://doi.org/10.34190/iccws.19.1.2086

Biography

Dr. Xiaolong Li is a professor in the Department of Electronics and Computer Engineering Technology at Indiana State University. He received his PhD in Computer Engineering from the University of Cincinnati in 2006. His primary areas of research include modeling and performance analysis of MAC protocol, Internet of Things, Wireless Ad Hoc networks, and sensor networks.

Dr. John Pickard is a professor of Information and Cybersecurity Technology at East Carolina University, North Carolina, USA. He received his PhD in Technology Management from Indiana State University in 2014. His main research areas are internet protocols, convergence of information and operations technologies, and Internet of Things applications.

Dr. Peng Li received his Ph. D. in Electrical Engineering from the University of Connecticut. His professional certifications include CISSP, RHCE and VCP. Dr. Li is currently an Associate Professor at East Carolina University. He teaches undergraduate and graduate courses in programming, computer networks, information security, web services and virtualization technologies. His research interests include virtualization, cloud computing, cybersecurity, and integration of information technology in education.