General Dynamics Mission Systems

Modeling the Compromise of a Network

using Bayesian Statistics

Our team performed Monte Carlo analysis of the vulnerabilities associated with networks using Microsoft 10, Mac OS and Linux operating systems. By obtaining data from the Common Vulnerabilities and Exposures (CVE) database the team created a probability matrix populated with the vulnerabilities associated with each operating system. Using Bayesian Statistics and linear algebra, we wrote an algorithm that modeled an attackers movement through a network using privilege escalation. With the help of high performance computing, we simulated networks of 30,000 nodes 1,000 times and recorded how quickly the networks were compromised in each simulation. We visualized this data with a histogram overlaid with a line indicating the distribution. Future work to improve the model will focus on inputting probability matrices that more accurately represent specific networks, and including security measures that protect the network as factors in our model.

Cyber Warriors

Matthew Risley, Patrick Seise, Elizabeth Archer

Virginia Tech

1 Problem Statement

Develop a model to assess the risk of compromise of a network via privilege escalation or lateral movement using privileged accounts, assuming the attacker is already in the network using a compromised unprivileged account.

2 Ethical Considerations

The team considered three possible ethical dilemmas associated with the model:

1. **Invalid Model** If a client either unnecessarily increases security or fails to increase security based off of the recommendation of invalid results from the model, the model is unethical.
2. **Incorrect Client Interpretation** If a client unnecessarily increases security or fails to increase security based off of an incorrect interpretation of the correct recommendation of our model, the model is unethical.
3. **Malicious Use** If an attacker discovers and utilizes the model to assess the vulnerability of networks to determine the most vulnerable network to attack, the model is unethical.

3 Literature Review

The team utilized three pieces of literature throughout the project.

The team utilized one piece of literature more closely than any others and that was *Network Vulnerability Assessment Using Bayesian Networks.* The contents of the piece of literature are described by its title. This piece of literature changed our project as we developed our math model based on its contents. Our mathematical model very closely resembles the mathematical model in the paper. [3]

Exiting the Risk Assessment Maze: A Meta-Survey, provided to the team by Dr. Colbert (our field trip contact), overviews cyber security risk assessment. This paper provides a high-level look at risk assessment and how analysts make assessments for a network. The team shall utilize this paper to understand risk assessment from a broader perspective without the specifics of data analysis [1].

A Cyber Attack Modeling and Impact Assessment Framework explores a possible cyber attack model as well as exploitation impact assessment. This paper relates to the project as it describes a set of algorithms for security evaluation, an aspect similar to what our math modeling component involves. The paper contains attack graphs and security metric calculations which serve as possible examples for our math modeling component [2].

4 Project Criteria

The team established three components. Each of the components is dependent on one another:

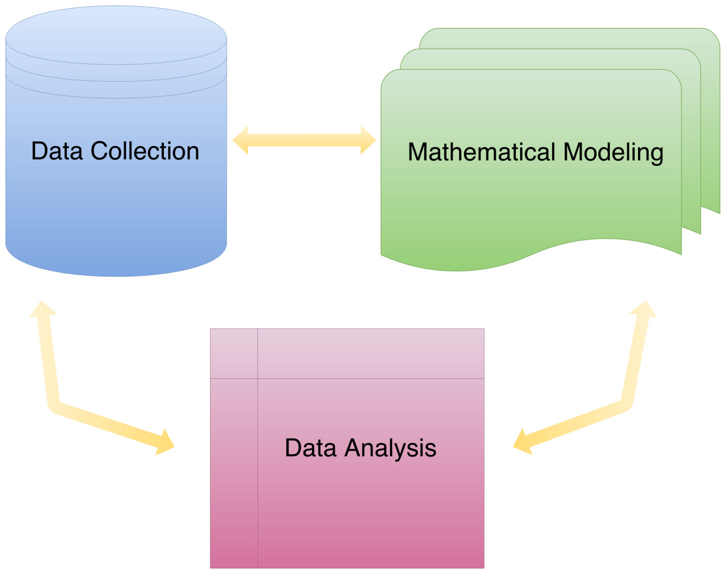


Figure 1: Project Components

For each component of the project, we establish a list of criteria that characterize an ideal solution. See below the list of criteria for each component:

Criteria for (1) Data collection.

- *Number of data sources.* Collection of 5-10 data sources, the range specified by the client.
- *Data format.* Data collected through webscraping and other means that exists in a usable file format (.csv, .txt, etc.).
- *Access.* Data collected through public sources that require zero additional cost.
- *Accuracy.* Data collected from reputable sources with 0 data falsification accusations.
- *Integration.* Data collected that varies no more than 50% from all other sources.
- *Ease.* Data collection that involves data that requires no more than 4 weeks to find.

Criteria for (2) Mathematical Modeling

- *Language.* Python or R, as requested by the client.
- *Speed.* The model shall take no longer than 24 hours to run on a 24 core super computer.
- *Accuracy.* The model should produce no more than 10 percent false positives.

- *Ease.* Simple usability and implementation given the current knowledge of the team and the single semester time constraint.

Criteria for (3) Data Analytics.

- *Language.* Language specified by the client. Develop in R and translate to Python if requested.
- *Visualization.* Visualization of data through spider charts that illustrate the importance of each factor.
- *Scope.* Analysis of data related to privileged account escalation.
- *Accuracy.* Over estimation of risk (false positives no more than 10% of the time) through statistical inferences.

5 Selected Solutions

Below are the assumptions of our Mathematical Model:

1. Each node in the network represents a privilege level of a machine on a network (computer, router, server, etc.).
2. Each edge of the network represents the probability that an attacker can escalate their privilege on that machine or move to another machine on the network.
3. Once a node is compromised, it will stay compromised.
4. An attacker can only attempt to compromise a node that is immediately connected to an already compromised node.

Given a network with k - nodes we can assume there is a compromised state vector $\underline{N} = \begin{bmatrix} X_1 \\ \vdots \\ X_k \end{bmatrix}$.

Each node within the network will be represented as x_i where $x_i \sim \text{Bern}(P_i)$

$$x_i \epsilon \{0,1\}, \text{ where } \begin{cases} 0 = \text{not compromised} \\ 1 = \text{compromised} \end{cases}$$

Each node has probability p_i of being compromised:

$$P(x_i = 1) = p_i$$
$$P(x_i = 0) = 1 - p_i$$

Each corresponding parent node will be denoted as Pnt_i, from this we can assume that the probability that a node will be compromised in the $n+1$ state as

$$P(x_i^{(n+1)} = 1 \mid \text{Pnt}_i^{(n)}) = 1 - \prod_j (1 - P(x_i^{(n+1)} = 1 \mid x_j^{(n)}), \text{ where } j \epsilon \text{Pnt}_i$$

6 Generalized Visualization

Below is an example of a small network with five nodes where node x_1 is compromised.

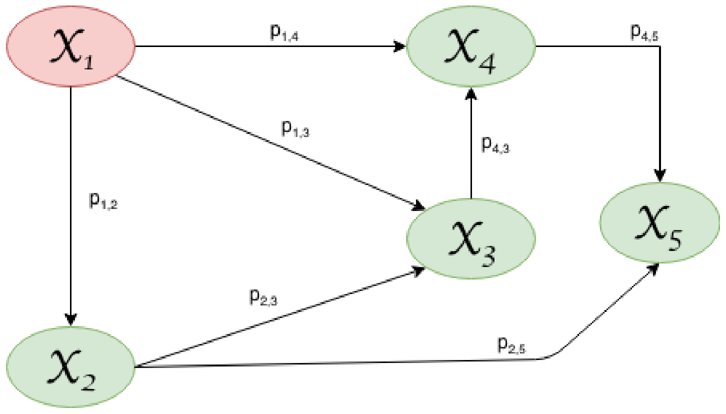


Figure 2: Network Visualization

The corresponding initial state vector is: $\underline{N} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

The probability that node x_j will be compromised through escalation privilege from node x_i is denoted as p_{ij}.

Based off of the example network we build the following matrix:

$$\begin{bmatrix} 1 & p_{12} & p_{13} & p_{14} & p_{15} \\ 0 & 1 & p_{23} & 0 & p_{25} \\ 0 & 0 & 1 & p_{34} & 0 \\ 0 & 0 & 0 & 1 & p_{45} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The network will always yield an upper triangular matrix because privilege escalation attacks imply a progressive movement through a network (i.e. node x_j will never point to node x_i).

We calculate p_i for the $n+1$ state below:

$p_1 = 1$
$p_2 = 1 - (1 - p_{12})$
$p_3 = 1 - (1 - p_{13})(1 - p_{23})$
$p_4 = 1 - (1 - p_{14})(1 - p_{34})$
$p_5 = 1 - (1 - p_{15})(1 - p_{25})(1 - p_{45})$

We perform Monte Carlo sampling using the p_i values. Once the values are calculated, we do a random sample to determine if node x_i is compromised at the subsequent time stage.

7 Results

The team has developed the project primarily in python. This includes data integration, matrix generation, simulation, and visualization. Focusing on collaboration the team has utilized a public git repository: `http://github.com/MattRisley/CyberWarriors_Capstone`. Functions have been developed in separate files allowing each member of the team to work on specific components of the project.

7.1 Small Simulation

When running through the teams simulation a matrix M is generated with dimension $n \times n$, where n can be specified by the user. The values contained within the matrix are generate from the CVC score data and placed randomly throughout. To emulate a realistic network the matrix is upper triangular with 1's on the main diagonal and CVC data randomly placed along the off diagonal as well as 30% of the remaining upper triangular matrix. The team aims to allow for minimal connections between nodes not adjacent to itself. The values inserted within the matrix represent which nodes are connected and the associated probability that the corresponding node will be exploited by an attacker.

$$\begin{bmatrix}
1. & 0.64 & 0. & 0.28 & 0. & 0. & 0.28 & 0.28 & 0. & 0. \\
0. & 1. & 0.28 & 0. & 0. & 0. & 0.28 & 0. & 0.07 & 0. \\
0. & 0. & 1. & 0.28 & 0. & 0. & 0.56 & 0. & 0. & 0. \\
0. & 0. & 0. & 1. & 0.28 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 1. & 0.28 & 0.28 & 0.28 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 1. & 0.28 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 1. & 0.56 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0.28 & 0.64 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0.28 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{bmatrix}$$

Each value represents the probability of the i^{th} to j^{th} node.

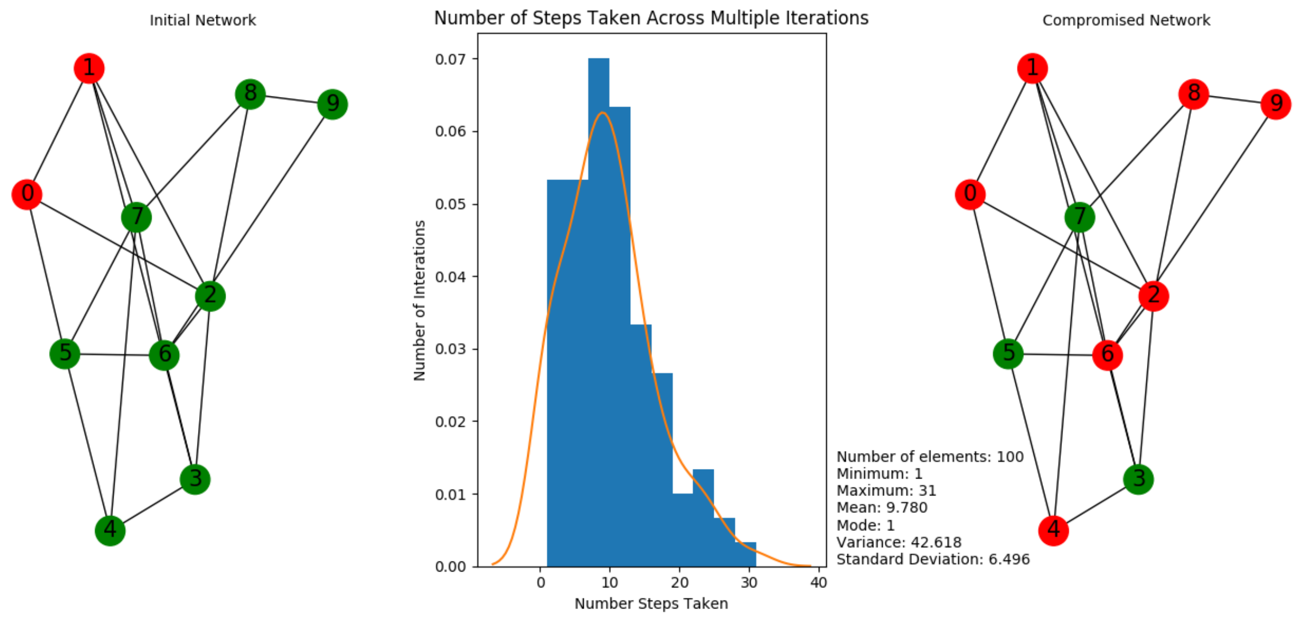
Upon running the simulation a few key mathematical computations occur in the background. Given the matrix M two addition vectors as generated. One indicating the initial compromised state of the network, denoted as **Net**. The team was previously told that the network should be assumed compromised and for the sake of simplicity the team as limited the scope to assume that the first two nodes within the network would be compromised.

$$\underline{\textbf{Net}} = \begin{bmatrix}1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

The second vector generated is the **watch** vector. This vector is randomly generated once and used for each iteration. The values within the vector are randomly placed as zero or ones indicating which nodes to *watch* for the network to become "fully" compromised. An example of this vector can be seen below.

$$\underline{\textbf{watch}} = \begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1\end{bmatrix}$$

To provide an adequate amount of variability the team as decided to allow flexibility on the number of iterations to run through the simulation. Utilizing a random draw technique the simulation generates a thresholds in-order to determine if the next node is likely to become compromised. This compromised vector can be seem above is on the right. Given a toy example, using a matrix M with size $n = 10$, 100 iterations will generate a distribution plot shown in Figure: 3



```
Number of elements: 100
Minimum: 1
Maximum: 31
Mean: 9.780
Mode: 1
Variance: 42.618
Standard Deviation: 6.496
```

Figure 3: Small Network Visualization

Before the simulation is run each node, except for the first two, are green. This represents an un-compromised state. Using `networkx`, a python library, the team is able to visualize the above network as a series of connected nodes.

This distribution informs us that for a network with 10 nodes with varying connections it would only take, on average, 9 steps to achieve the compromised state defined by the randomly generated **watch** vector.

7.2 Scaling Up

As the team scaled up the simulation optimizing for performance was a key focus. Using the `multiprocessing` library the team was able to incorporation parallel processing across iterations. In an effort to simulate a realistic corporate network we defined the size of our M matrix with n = 30,000. Given the sheer size of this network, it was realized to be computationally intensive. Utilizing ARC's performance capabilities the team was able to distribute the computational tasks across multiple processors.

The team uncovered vulnerability scores relating to Windows 10 and macOS systems. Given that the CVC scores are stored as a difficulty level the team converted this information is probabilistic terms using the below formula.

$$p_{ij} = 1 - (\text{score} \cdot 0.10)$$
where i and j represent the position in the network matrix

Utilizing this information to populate a matrix the simulation produced the following distribution across the variety of steps taken to achieve a fully compromised state.

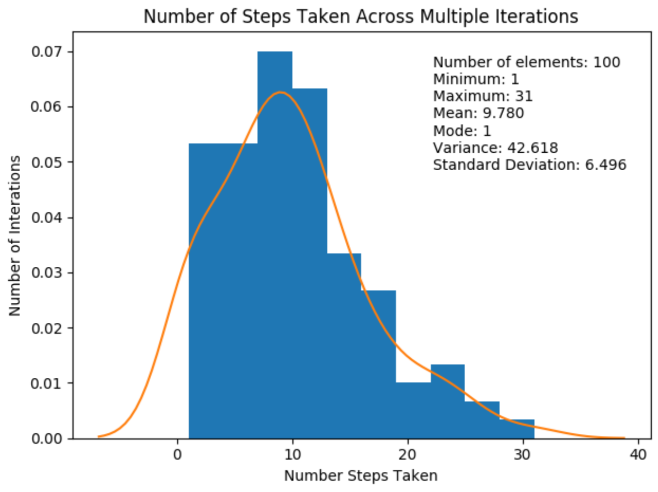


Figure 4: Windows 10 Network Visualization

Both the Windows 10 and macOS simulations generates similar results.

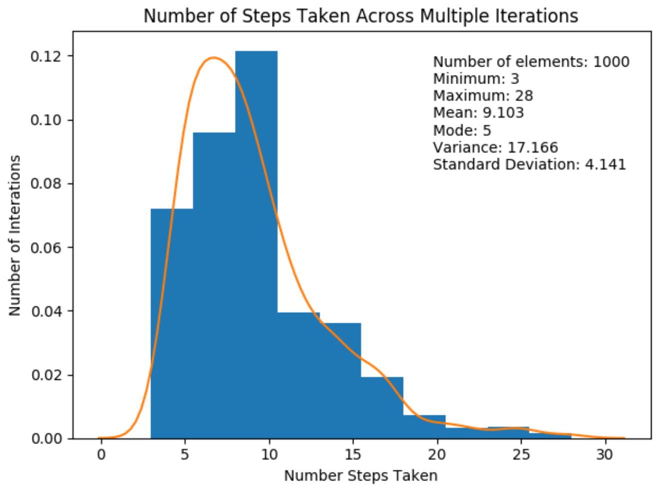


Figure 5: macOS Network Visualization

This might be due to the sheer size of the networks and the variety of probabilistic terms spread throughout the network.

8 Obstacles

1. **Data Collection** The team struggled to find data during the early stages of our model. The team was originally asked by our client to use the following factors in our model: time privileged account passwords left unchanged, time account active, usage of the account, number of hosts/applications with same credentials, total number of hosts and applications in the network, criticality of the application, and levels of account privilege. However, no data of that kind exists, either open source, or provided by our client. The team overcame this obstacle by using the data on the Common Vulnerabilities and Exposures Database to determine the probabilities used in our model.

2. **Markov Chains** The team first decided to utilize Markov Chains for our implementation. However, our implementation of Markov Chains would not have had a real world application because it assumed that an attacker would only be on one node at a time, as it described each node as a "state" in the Markov chain. The team overcame this obstacle by finding a different approach and instead implementing the Bayesian network model.

3. **Learning the Math** The team decided to utilize Bayesian Networks for our model. However, the team had to take the time to learn the Bayesian statistics related to our model in order to implement it. The team overcame this by working with our coach to learn the math required in a reasonable amount of time.

4. **Computation** The team wanted to test the model on a very large network. However, on a standard personal computer, running a large network is impractical. The team overcame this obstacle by using advanced research computing resources provided by the university (i.e. New River).

9 Roles

All:
The team as spent time with Rom in an effort to understand the mathematics behind the Bayesian mathematical model. Each member has put in the time and effort to learn the components needed to program, analyze, and visualize the model.

Patrick:
Collected and cleaned the CVE security score data using web-scraping techniques. Developed some of the helper functions to watch for our compromised state for the entire network.

Matthew:
Set up git repository for collaboration. Tested and wrote portions of the simulation python file to get everything up and running. This also includes other helper functions like generating random matrices and state vectors for the simulation.

Elizabeth:
Discovered the crucial CVE data needed to provide accurate probabilistic data to the generated networks. Developed the python script to calculate the probability for the next state in the simulation along with determining if the adjacent node will become compromised with the next interaction.

10 Conclusions and Future Work

Through our work we learned that configuring a network so that the connectivity between the nodes is layered can make the network more secure. That is, a network where all nodes are connected is less secure than a network where each node is connected to only a select few other nodes. While this may be inconvenient for the network owner, should the network get infiltrated by an attacker a layered network would slow the attacker's progress by forcing them to go through more nodes in order to reach their target node. Increasing the amount of time it takes an attacker to compromise a network improves the chances of detecting the attacker and protecting any sensitive information on the network.

For future work our team wants to create a network that more accurately represented our clients network in terms of operating systems, the level of connectivity between the nodes, and expanding vulnerabilities beyond privilege escalation. Once this is done the team would factor in common security measures such as firewalls, and password requirements to get a realistic understanding of how these systems impact an attackers ability to traverse through a network. Lastly, the team would improve upon our algorithm by record the specific path the attacker would likely take and optimize the efficiency of each iteration of the simulation.

References

[1] D. Gritzalis, G. Iseppi, A. Mylonas, and V. Stavrou, *Exiting the risk assessment maze: A meta-survey*, ACM Comput. Surv., 51 (2018), pp. 11:1–11:30.

[2] I. Kotenko and A. Chechulin, *A cyber attack modeling and impact assessment framework*, (2013), pp. 1–24.

[3] H. M. Yu Liu, *Network vulnerability assessment using bayesian networks*, tech. rep., SPIE, March 2005.