Date:	16 November 2018
To:	Romcholo Macatula
From:	Cyber Warriors
Subject:	Technical Memo 6: Preliminary Results

1 Summary

In this memo, we discuss our preliminary results and each team member's contribution to those results, our original gannt chart in comparison to our updated gannt chart, and our biggest challenge thus far in our project. We preface our preliminary results with a breakdown of the math behind our Bayesian statistics model of a network and show an example with a small network using the model. Our preliminary results discuss the code that was written to implement our model in the python programming language. We discuss the matrix and vector parameters that our model take in and we give an example of each. We also show visualizations from the `networkx` package in python. We also show an example graphical distribution of our simulation run with 100 iterations. Following our preliminary results, we display our gannt charts and discuss the reasoning behind adding in the additional Data Analysis component to our project. Finally, we discuss the biggest challenges we have faced with the project, those being the understanding of the mathematics behind the Bayesian statistics model and scope definition.

2 Mathematical Model

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 $x_i \sim \text{Bern}(P_i)$

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

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

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

3 Generalized Visualization

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

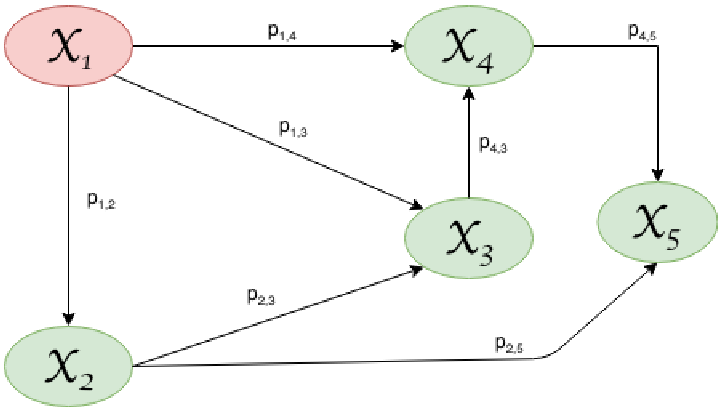

Figure 1: Network Visualization

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.

4 Preliminary Results

The team has developed the project extensively in python. This includes data integration, matrix generation, simulation, and visualization. Focusing on collaboration the team has implemented a public git repository: `https://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.

When running through the teams simulation a matrix M is generated with dimensions $n \times n$. The values contained within the matrix are generated at random between 0 and 1. These values are inserted on the diagonal to represent which nodes are connected and the associated probability that the corresponding node will be exploited by an attacker.

$$\begin{bmatrix} 1 & 0.95 & 0.54 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1 & 0.61 & 0.73 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1 & 0.29 & 0.47 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1 & 0.49 & 0.6 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1 & 0.44 & 0.01 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1 & 0.48 & 0.23 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1 & 0.76 & 0.18 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1 & 0.86 & 0.26 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1 & 0.8 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1 \end{bmatrix}$$

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

Using `networkx`, a python library, the team is able to visualize the above network as a series of connected nodes. Before the simulation is run each node is green representing an uncompromised state. The team is stilling exploring various visualization packages that allows for more versatility in adding in the probabilities between nodes along with reorienting the visual structure.

Upon running the simulation a few key mathematical computations occur in the background. Given the matrix M two addition vectors are generated. One indicating the initial compromised state of the network, denoted as $\underline{\mathbf{N}}$. Been previously told that the network should be assumed compromised the team limited the scope and for the sake of simplicity the network is initialized with the first two nodes identified as being compromised.

$$\underline{\mathbf{N}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The second vector generated is the $\underline{\mathbf{watch}}$ vector. This vector is randomly filled with zero or ones indicating which nodes to watch for the network to become "fully" compromised. Each simulation generates a random $\underline{\mathbf{watch}}$ vector as displayed below.

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

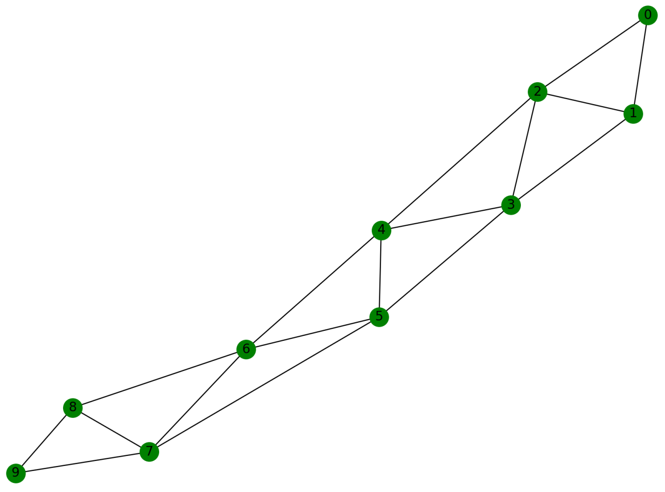

Figure 2: Uncompromised Network

To provide an adequate amount of variability the team as decided to allow flexibility on the number of iterations to run through the simulation. Given a toy example, using a matrix M with size $n = 10$, 100 iterations with generate a distribution plot shown in Figure: 3

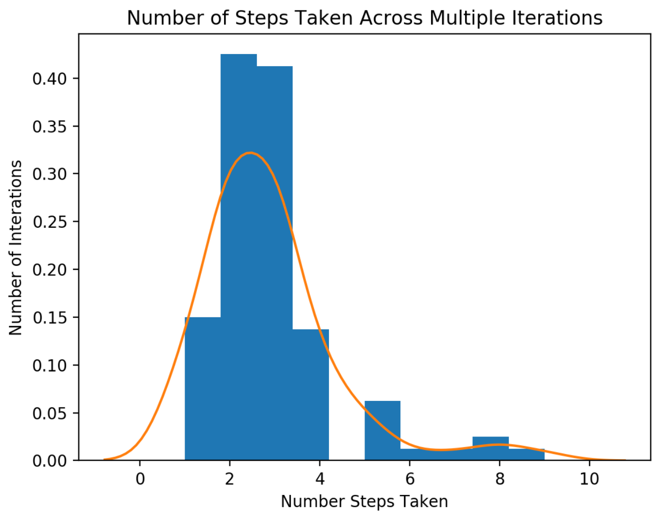

Figure 3: 100 iterations of simulation

This distribution informs the us that for a network with 10 nodes with varying connections would only take 6 steps to achieve the compromised state defined by the randomly generated **watch** vector. Upon achieving a fully compromised network, as defined by the **watch** vector, the network will turn red.

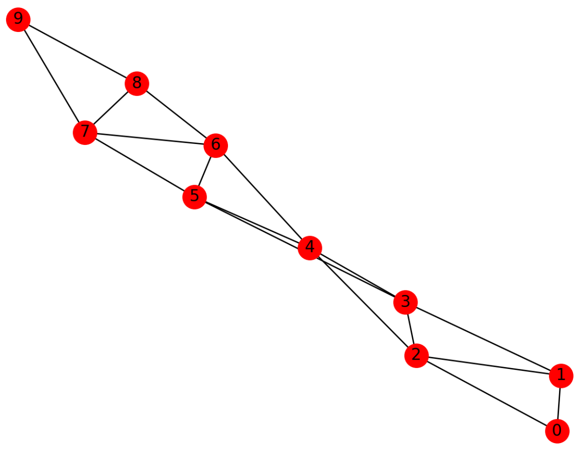

Figure 4: Compromised Network

5 Team Member Contributions

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.

6 Gannt Chart

The team discovered a large portion of the teams time should be spent generating a Monte Carlo style simulation for analyzing a compromised network. Focusing on drawing mathematical and statistical conclusions from the simulation, the team has sorted the initial data collection and cleaning phase. The team needed only the first two weeks to collect and clean the CVC security score data. The team plans to incorporate those probabilities into the generation of the simulation.

Original Gantt Chart

	O4	O11	O18	O25	O1	N8	N15	N22	N29	D5
Data Collection	▨	▨	▨	▨	▨	▨				
Data Cleaning	▨	▨	▨	▨	▨	▨	▨			
Math Modeling		▨	▨	▨	▨	▨	▨	▨	▨	
Presentation							▨	▨	▨	▨
Report							▨	▨	▨	▨

Updated Gantt Chart

	O4	O11	O18	O25	O1	N8	N15	N22	N29	D5
Data Collection	▨	▨								
Data Cleaning	▨	▨								
Math Modeling		▨	▨	▨	▨	▨	▨	▨	▨	
Data Analysis	▨	▨	▨	▨	▨	▨	▨			
Presentation							▨	▨	▨	▨
Report							▨	▨	▨	▨

The team seems to be on track with a few final components left to add within the mathematical modeling section.

7 Biggest Challenge

The biggest challenge the team has faced so far is understanding the Bayesian statistics along with narrowing the scope to a Monte Carlo style simulation. The next big steps the team plans to take involve generating matrices using the CVE scores along with using the computational power of newriver to scale the teams simulation to model realistically sized networks.