INTRUSION DETECTION SYSTEM USING A NEURAL NETWORK

PHASE ONE PRESENTATION

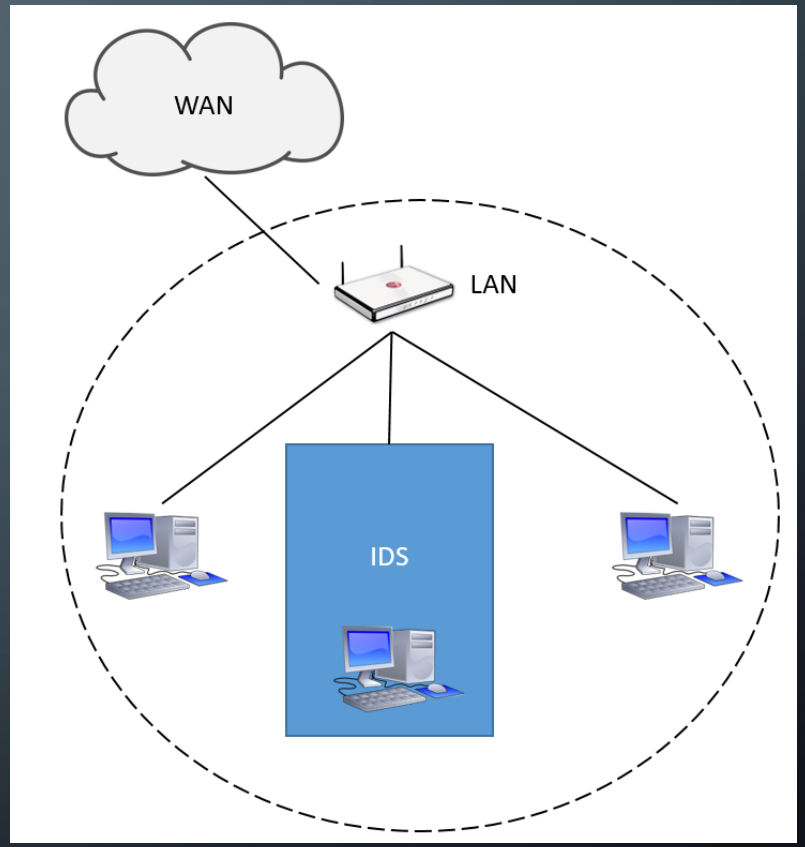
BLAKE KNEDLER

AGENDA

- Project Vision and Overview
- Project Requirements
- Project Plan
- Cost Estimation using COCOMO
- Architecture Elaboration Plan (Phase Two)
- Software Quality Assurance Plan
- Risks and Concerns
- Alpha Prototype Demonstration
- Questions and Comments

PROJECT VISION AND OVERVIEW

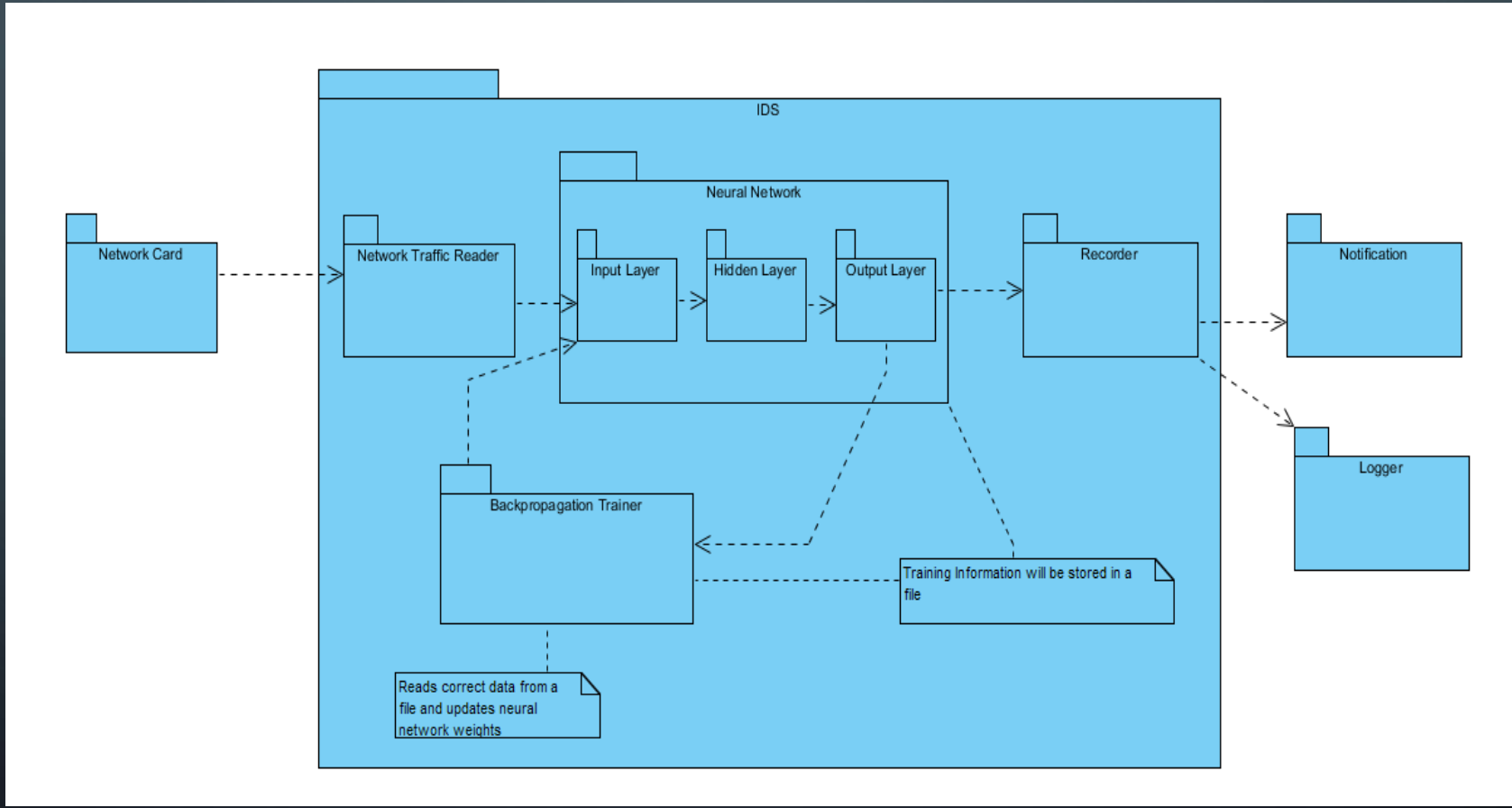
- Intrusion Detection System
  - Host-based system
  - Read TCP/IP Packets
  - Neural Network Decision Making
  - Notification System
  - Logging System



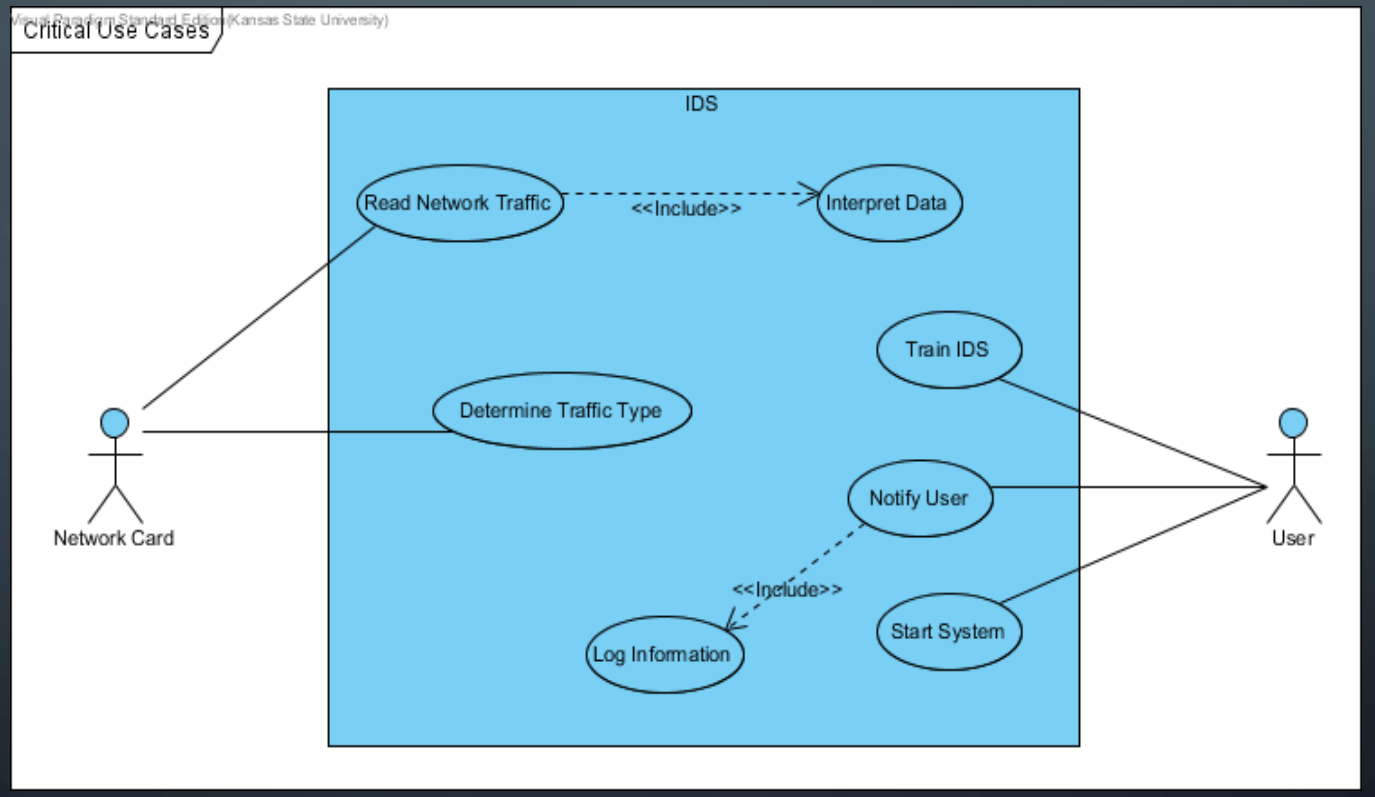
PROJECT VISION AND OVERVIEW

- Goals for the Intrusion Detection System
  - Create a TCP/IP layer to read all packets received by host system
  - Create a Neural Network layer to make decisions if the packet is a DOS attack
    - Neural Networks are not perfect, aim for 90% accuracy
  - Create a notification system for when a malicious packet is received
- Motivation
  - Create a tool to help prevent DOS attacks
  - Allow for future work to completely prevent DOS attacks when noticed

PROJECT VISION AND OVERVIEW



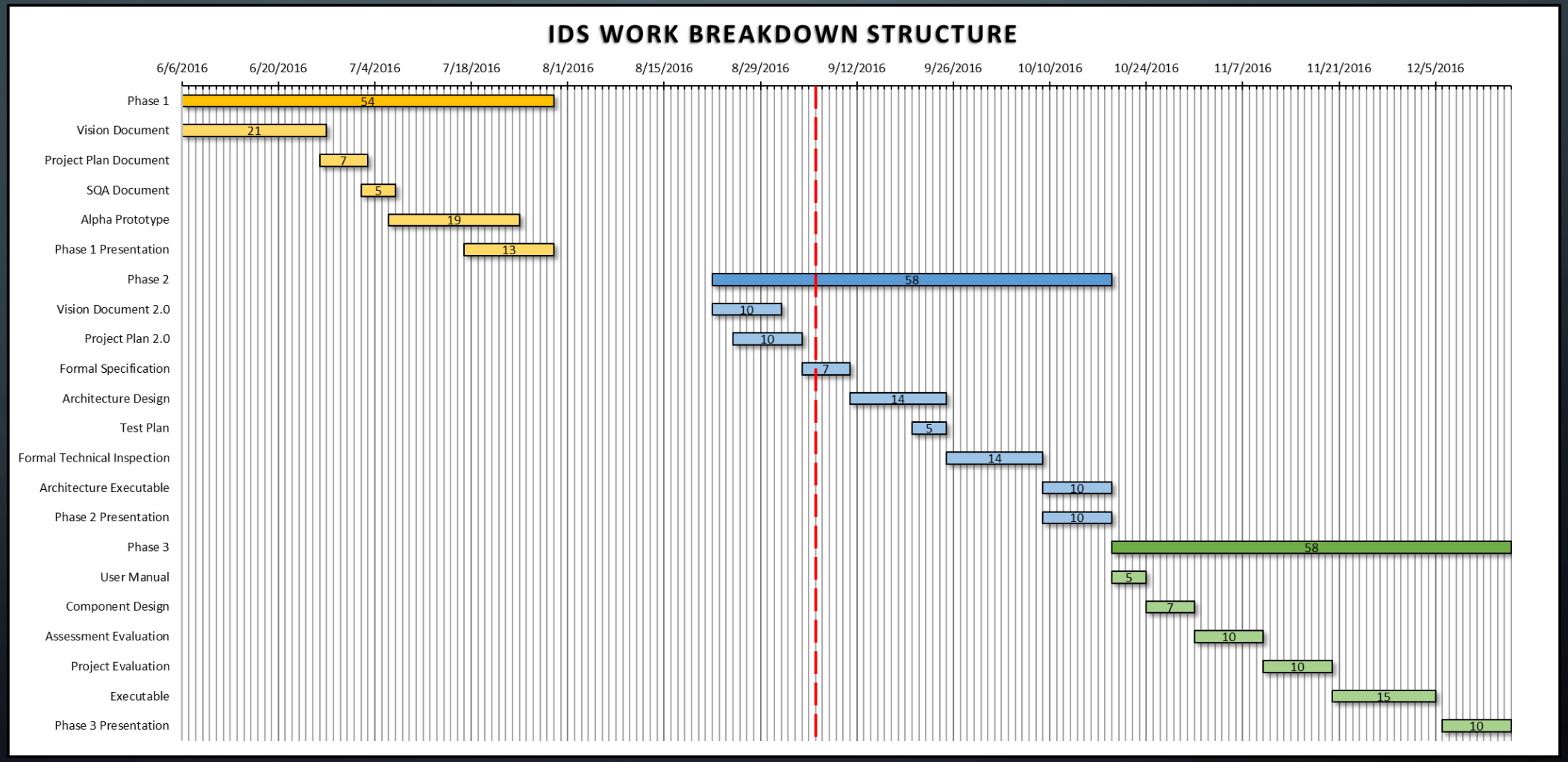
PROJECT REQUIREMENTS



PROJECT REQUIREMENTS

- SR1.1: [CR] The System shall be able to read data from the host system network card.
- SR2.1: [CR] The system shall be able to interpret the data from the received network traffic and store it in a usable format.
- SR3.1: [CR] The system shall be able to determine if the network data received by the host machine is malicious with at least 85% accuracy.
- SR3.2: The system shall determine what type of attack is being made to the host network when malicious network traffic is found.
- SR4.1: [CR] The system shall be able to train itself through backpropagation on know network traffic data.
- SR5.1: [CR] The system shall be able to notify the User of the host system when a malicious attack is encountered.
- SR6.1: The system shall be able to log all malicious attacks into a log file.

PROJECT PLAN



COST ESTIMATION USING COCOMO

Name	Formula	Result	Units
Effort Applied (E)	a*(KLOC)^b*EAF	**12.82**	**Man-Months**
Development Time (D)	c*(E)^d	**6.59**	**Months**
People Required (P)	(E) / (D)	1.94	People Count

Cost Drivers	Ratings						Chosen Result
	Very Low	Low	Nominal	High	Very High	Extra High	
Product attributes							
Required software reliability	0.75	0.88	1	1.15	1.4		1.15
Size of application database		0.94	1	1.08	1.16		1
Complexity of the product	0.7	0.85	1	1.15	1.3	1.65	1.3
Hardware attributes							
Run-time performance constraints			1	1.11	1.3	1.66	1.3
Memory constraints			1	1.06	1.21	1.56	1
Volatility of the virtual machine environment		0.87	1	1.15	1.3		1
Required turnabout time		0.87	1	1.07	1.15		1.07
Personnel attributes							
Analyst capability	1.46	1.19	1	0.86	0.71		0.86
Applications experience	1.29	1.13	1	0.91	0.82		0.91
Software engineer capability	1.42	1.17	1	0.86	0.7		0.7
Virtual machine experience	1.21	1.1	1	0.9			1
Programming language experience	1.14	1.07	1	0.95			0.95
Project attributes							
Application of software engineering methods	1.24	1.1	1	0.91	0.82		1
Use of software tools	1.24	1.1	1	0.91	0.83		0.83
Required development schedule	1.23	1.08	1	1.04	1.1		1.04
TOTAL EAF	**0.9342**						

ARCHITECTURE ELABORATION PLAN

- Revise Vision Document
- Revise Project Plan
- Create Formal Specification
- Create Architectural Design
- Create Test Plan
- Conduct Technical Inspection
- Create Executable Architecture Prototype

SOFTWARE QUALITY ASSURANCE PLAN

- Management
- Software Reviews
- Testing
- Problem Reporting
- Tools, Techniques, and Methodologies
- Revision Control

RISKS AND CONCERNS

- Correctly reading all information needed from the TCP/IP packets quickly
- Quickly reading all the packet information into the neural network
- Tweak the neural network as necessary in the attempt to get as many correct answers as possible.

ALPHA PROTOTYPE DEMONSTRATION

- GitHub Repository Location:
  - https://github.com/bneedy/PyIDS
- Worked on the most complex part of the system, the neural network.
- Aimed at training the system with a few hundred messages and see how it handled a larger dataset.
- Had around 90-95% accuracy on around 450,000 messages training the system with around 200 messages.

QUESTIONS AND COMMENTS