Final Report for Preventing Black Hole Attacks in MANETS Using Machine Learning

Audience:
Project sponsors, project mentors, unit coordinators and all others in the university and ITC community interested in this project.

Purpose

The purpose of this document is to clarify the scope, methodology and objectives of this capstone project. The main stakeholders should evaluate the final report and provide feedback and suggestions that could improve any future work.

Authors	Version	Date
Thi Doan, Alan Gaugler, Tristan Hore, Johnson Sangah	1.0	3 May 2023

Approval by Team Members:

Team Member Name	Student ID
Thi Doan	u3090674
Alan Gaugler	u885853
Tristan Hore	u3218682
Johnson Sangah	u3082492

Table of Contents

1. Introduction and Description of the Project

A Mobile Ad Hoc Network (MANET) is a type of wireless network where several devices, such as radios, smartphones, laptops, cars, drones or sensors, communicate directly without needing any pre-existing or centralized infrastructure. MANETs are self-configuring and self-maintained by their nodes, making them ideal for situations where existing network infrastructure is damaged, unavailable, or not possible such as in disaster-stricken areas, military operations, or remote regions.

For example, in a region recently devastated by an earthquake, communication infrastructure such as mobile networks may be severely damaged. In such an event, rescue and emergency services could rapidly set up a MANET to quickly establish a wireless network among their devices to exchange vital information and communications. Likewise, in military operations where fixed network infrastructure is unavailable, MANETs can enable military personnel and unmanned devices to communicate and share tactical information in real time using wireless devices.

A commonly used routing protocol in MANETs is the Ad-hoc On-Demand Distance Vector (AODV) protocol. AODV is a reactive routing protocol, which means that it establishes routes between nodes on-demand when required rather than maintaining pre-established routes. AODV uses a route discovery mechanism where nodes broadcast route requests to find a route to a destination node. When a route is discovered, AODV establishes a temporary route and maintains it for as long as the nodes are actively communicating. AODV protocol is widely used in MANETs due to its simplicity, scalability, and adaptability to changing network topologies due to the mobility of nodes.

However, protocols like AODV are vulnerable to various types of cyber-security threats, one of the most common being black hole attacks. A black hole attack is a type of cyber-attack where a malicious node infiltrates the network and falsely claims to have the shortest route to a destination, making it the prioritized path for link establishment. Doing so attracts routing and data traffic towards it, but it drops the received packets instead of forwarding them to the desired destination. This will severely disrupt communication in the network between nodes and cause the loss of vital data, leading to performance degradation and network instability. Hence, black hole attacks can cause severe disruptions in critical operations, making them a significant security concern in MANET environments.

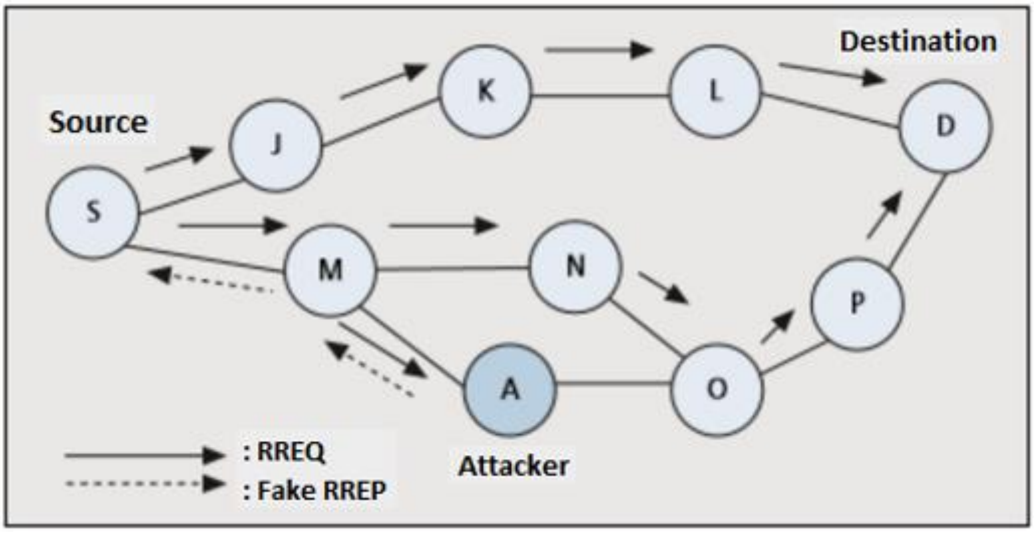


Figure 1 – Diagram of MANET with a Black Hole Node A (Source: Fakultas et al.)

Figure 1 shows a diagram of a MANET where the source node S sends a route request to the destination node D. The malicious node A replies with a fake route response message, claiming to have the shortest route to node D. Node S would then establish a route to node D through node A, but the packets would be dropped by node A instead of being forwarded to the destination.

This project aimed to develop a solution using machine learning techniques that could detect black hole intruders in a MANET utilizing the AODV protocol. The project consisted of three major phases.

Phase 1

In the data creation phase, a MANET network would be simulated using the NS3 simulator, incorporating the AODV routing protocol. The simulation would also include a small number of black hole nodes. Finally, the simulation would generate output files comprising various network metrics, AODV messaging and data flow between the nodes.

Phase 2

In the data preparation phase, a script would be developed to process the simulation files and convert them into a dataset that could be used to train a machine learning classifier. This would involve identifying and extracting various relevant features that could be used to detect the behaviour of black hole nodes.

Phase 3

The machine learning (ML) phase would develop a machine learning process using the dataset generated in stage 2 as input to train ML classifiers to detect which neighbour nodes were black hole nodes.

2. Summary of Outcomes

Phase 1 - data creation phase: a NS3 simulator was used to simulate MANET networks using the AODV protocol. Three scripts were developed in C+ to simulate the interaction of all nodes in the MANET. Black hole nodes were also introduced into the network. The NS3 scripts used were able to produce simulations of standard AODV networks. However, due to the complexity of the process, the tracing configured in the script was not producing complete logs in the AODV with black hole node simulations. This made manually altering the messaging logs necessary to enable proof of concept work on training the ML models. Thus it cannot be said that we produced succesful blackhole simulations of the full network. Despite these difficulties, the standard AODV scripts output and toy Blackhole examples produced good output that could still be used for the project's next phases.

Phase 2 - data preparation phase: the literature review identified the key behavioural characteristics of black hole nodes in AODV networks. From this, a complex program was developed that would read the AODV protocol messaging between nodes from the trace files produced in the previous phase. The script would then extract key features from the messaging from which a dataset was created that could be used in the machine learning process. This phase was successful, and datasets could be produced from the trace files.

Due to the issues discussed above, some values of the feature set of the black hole nodes were manually altered in the training set to accurately portray the behavioural characteristics of a black hole node. With this step, the project could continue.

Phase 3 - machine learning phase: a machine learning script was developed to detect black hole nodes within the network. The datasets created by the data conversion script developed in the Data Preparation Stage were used as the input. The variables with the highest correlation to the target variable were determined and they were filtered into the final dataset for model training. Two binary classification models were developed, a random forest classifier and a support vector machine classifier. Both models could classify with 100% accuracy which nodes were black hole nodes.

More work must be done to get the NS-3 simulator to accurately simulate AODV networks and black hole nodes. Afterwards, more training and testing of the models is required. The *opportunities for future development* section mentions what steps should be carried out next. However, with all the hard work put into this project, the project's final phase was successfully completed. The results demonstrate that machine learning can successfully be applied to AODV networks to detect black hole nodes accurately.

3. Project Deliverables Report

In the initial scope of the project, six major deliverables were defined. This section reports on the processes and outcomes of each of these deliverables.

Literature review

The team carried out an extensive literature review of MANETs, AODV protocol and typical black hole cyber-attacks. This in-depth review was essential in correctly understanding how nodes in MANETs, in particular those that employ the AODV protocol, behave and establish communication. It was also essential to understand how a black hole node would infiltrate the network and interact with other nodes, acting as a transfer point between nodes but actually dropping the data packages and thereby disrupting essential communications. The team also viewed many videos, which proved essential in learning the protocols and configuration of the NS-3 network.

The literature review gave us a comprehensive understanding of MANETs, the AODV network protocol, and the behavioural characteristics of the black hole nodes infiltrating the network. Several of these characteristics could be used for a network node to identify a first-tier neighbour as a black hole node.

Identifying these principal unique behavioural characteristics of black hole nodes, led to the development of a dataset that could be used for the machine learning process, to identify a malicious node.

Please refer to the literature review in the project deliverables for a more extensive description of the identified behavioural characteristics of black hole nodes and a bibliography of the most informative and relevant research papers reviewed for this project.

Source code for the NS-3 simulations

The Network Simulator 3 (NS3), is an open-source network simulation tool developed by the NS3 community. Originating in 2006 as an evolution of the NS-2 simulator, NS3 provides a comprehensive platform for researchers and educators to design, prototype, and analyze networks. NS3's modular architecture supports various network protocols, traffic models, and routing algorithms, this enables users to simulate complex networks. The simulator is widely used in academia to study wireless, wired, and hybrid networks and test and validate new networking technologies.

In this project, three scripts modified from the standard MS3 modules in line with existing black hole simulation examples [1][2][3], were used in the preparation of the NS3 simulation, along with many standard NS3 modules and tools. In addition, a simple blackhole model was used early on to understand network behaviour, providing a foundation for anticipating the behaviour in a more extensive network. [4]

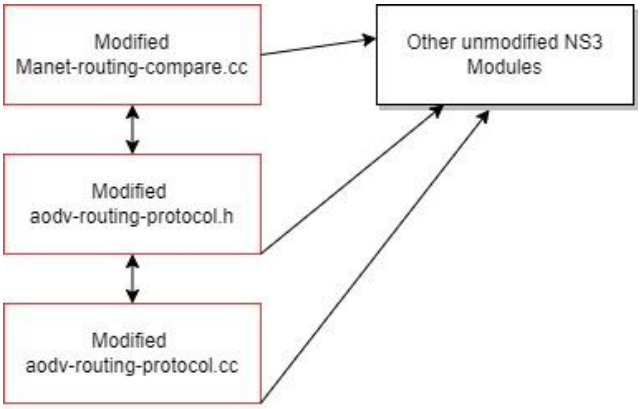


Figure 2 – Three modified scripts used to simulate the effects of a blackhole attack on an AODV network

In the modified implementation, changes are made to the behaviour of the AODV routing protocol when a node is marked as malicious:

When a malicious node receives a packet, it simply drops it instead of forwarding it. When a malicious node receives a route request (RREQ) message, it creates a false routing table entry with a higher sequence number and a lower hop count, effectively advertising itself as having the shortest path to the destination. The malicious node then sends a route reply (RREP) message to the source node, causing the source node to route its packets through the malicious node.

These modifications allow the simulation of blackhole attacks in a NS-3 network, enabling the study of the impact of such attacks on network performance and developing countermeasures to protect the network.

Changes in the NS3 simulator between the time of the writing of the manet-routing-compare.cc script and the NS3 installation used in this project caused several issues and required substantial debugging in addition to the changes needed to implement the blackhole simulations. In addition, further changes were required to match the changes in the AODV-Routing-Protocol.cc and AODV-Routing-Protocol.h scripts written for NS.3.25 with the versions that were included in the current installation used for the project. See the Lessons Learned and on Outstanding Issues for further details on these travails. The project closure report includes a further detailed explanation of the scripts' function and their use.

Data traces produced from the NS-3 simulations

The NS3 simulator provides a trace system that captures various simulation parameters and data and exports to various file formats using pre-built helper functions. The initial plan was to produce out in the widely used pcap format. However, we experimented with producing both flat ascii output and using the built-in flowmon functions, which is the preferred method in the latest simulator version. Finally, we concluded that the pcap files were the easiest to work with using external tools such as Wireshark, and so this was adopted as the primary output format for the network traces. In generating the PCAP files we encountered two sets of related issues.

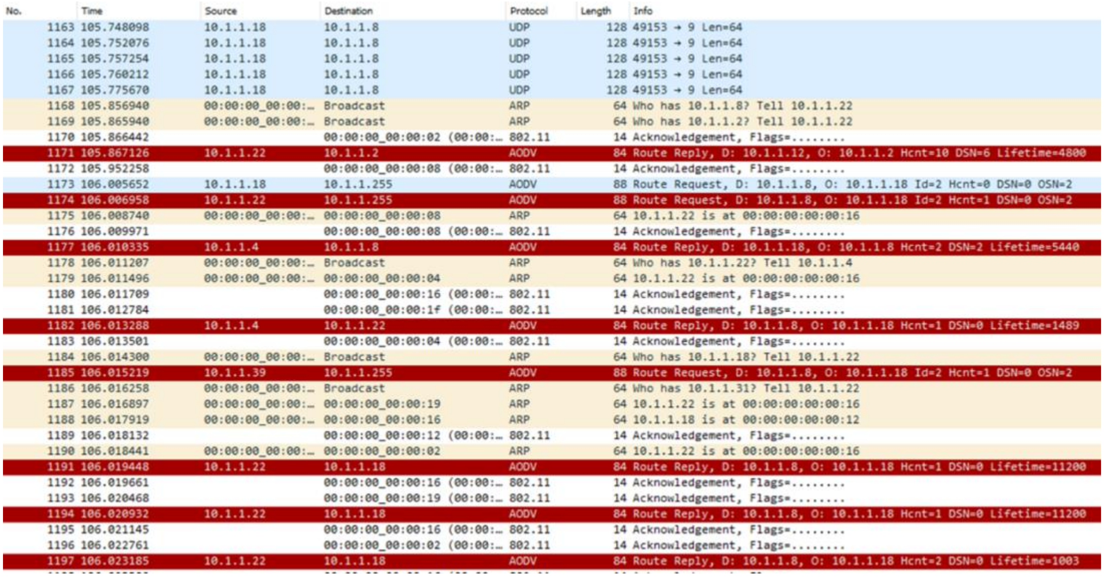
Firstly, the output files of the PCAP traces were produced from inside a loop n order that each node could be treated separately and to prevent the JSON files produced in the later steps from becoming unwieldy. However, this setup caused some strange behaviour in the log files, making it harder to follow all of the message traffic. Workarounds to this were developed for the data used to train the ML models, however, these issues were not fully resolved in the simulation scripts. Secondly, substantial difficulties in extracting the details of the modified attribute set on nodes to signify if the node was malicious, caused delays.

This issue eventually frustrated the fulfilment of the project's stretch goals. Initially, these issues seemed to be due to changes in the logging components between versions, however, this issue may be related to the way the changes to the script were structured, and work is continuing to resolve this issue by changing the way node ids are assigned.

Dataset Preparation

The next phase of the process was to create a dataset that could be used for the machine learning phase. The literature review identified key behavioural characteristics of black hole nodes in AODV networks. These were incorporated into the dataset as key features to be used in the machine learning process for the detection of black hole nodes.

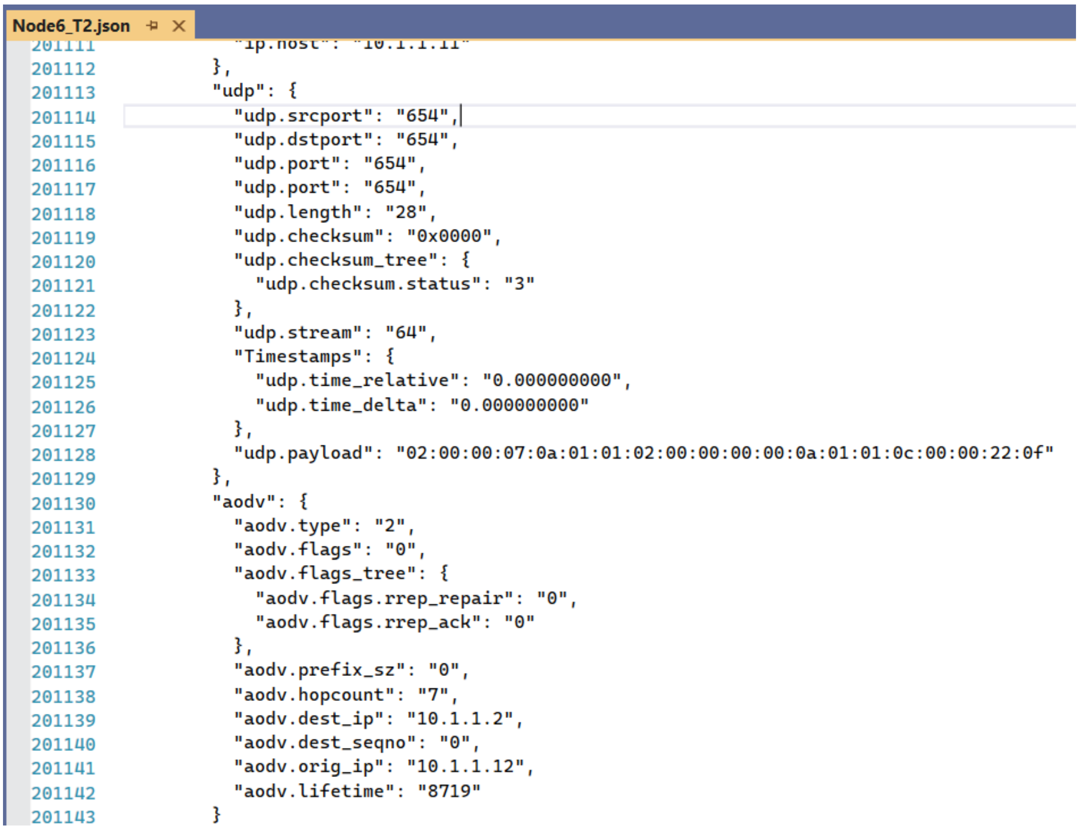
The data is created from running simulations, so collecting sufficient data for training and testing machine learning models is not an issue. The output of the trace files consisted of messaging files from each node. These were in pcap file format and loaded into Wireshark, where they could be observed. Wireshark is a free and open-source network protocol analyzer.



No.	Time	Source	Destination	Protocol	Length	Info
1163	105.748098	10.1.1.18	10.1.1.8	UDP	128	49153 → 9 Len=64
1164	105.752076	10.1.1.18	10.1.1.8	UDP	128	49153 → 9 Len=64
1165	105.757254	10.1.1.18	10.1.1.8	UDP	128	49153 → 9 Len=64
1166	105.760212	10.1.1.18	10.1.1.8	UDP	128	49153 → 9 Len=64
1167	105.775670	10.1.1.18	10.1.1.8	UDP	128	49153 → 9 Len=64
1168	105.856940	00:00:00_00:00:…	Broadcast	ARP	64	Who has 10.1.1.8? Tell 10.1.1.22
1169	105.865940	00:00:00_00:00:…	Broadcast	ARP	64	Who has 10.1.1.2? Tell 10.1.1.22
1170	105.866442		00:00:00_00:00:02 (00:00:…	802.11	14	Acknowledgement, Flags=........
1171	105.867126	10.1.1.22	10.1.1.2	AODV	84	Route Reply, D: 10.1.1.12, O: 10.1.1.2 Hcnt=10 DSN=6 Lifetime=4800
1172	105.952258		00:00:00_00:00:08 (00:00:…	802.11	14	Acknowledgement, Flags=........
1173	106.005652	10.1.1.18	10.1.1.255	AODV	88	Route Request, D: 10.1.1.8, O: 10.1.1.18 Id=2 Hcnt=0 DSN=0 OSN=2
1174	106.006958	10.1.1.22	10.1.1.255	AODV	88	Route Request, D: 10.1.1.8, O: 10.1.1.18 Id=2 Hcnt=1 DSN=0 OSN=2
1175	106.008740	00:00:00_00:00:…	00:00:00_00:00:08	ARP	64	10.1.1.22 is at 00:00:00:00:00:16
1176	106.009971		00:00:00_00:00:08 (00:00:…	802.11	14	Acknowledgement, Flags=........
1177	106.010335	10.1.1.4	10.1.1.8	AODV	84	Route Reply, D: 10.1.1.18, O: 10.1.1.8 Hcnt=2 DSN=2 Lifetime=5440
1178	106.011207	00:00:00_00:00:…	Broadcast	ARP	64	Who has 10.1.1.22? Tell 10.1.1.4
1179	106.011496	00:00:00_00:00:…	00:00:00_00:00:04	ARP	64	10.1.1.22 is at 00:00:00:00:00:16
1180	106.011709		00:00:00_00:00:16 (00:00:…	802.11	14	Acknowledgement, Flags=........
1181	106.012784		00:00:00_00:00:1f (00:00:…	802.11	14	Acknowledgement, Flags=........
1182	106.013288	10.1.1.4	10.1.1.22	AODV	84	Route Reply, D: 10.1.1.8, O: 10.1.1.18 Hcnt=1 DSN=0 Lifetime=1489
1183	106.013501		00:00:00_00:00:04 (00:00:…	802.11	14	Acknowledgement, Flags=........
1184	106.014300	00:00:00_00:00:…	Broadcast	ARP	64	Who has 10.1.1.18? Tell 10.1.1.22
1185	106.015219	10.1.1.39	10.1.1.255	AODV	88	Route Request, D: 10.1.1.8, O: 10.1.1.18 Id=2 Hcnt=1 DSN=0 OSN=2
1186	106.016258	00:00:00_00:00:…	Broadcast	ARP	64	Who has 10.1.1.31? Tell 10.1.1.22
1187	106.016897	00:00:00_00:00:…	00:00:00_00:00:19	ARP	64	10.1.1.22 is at 00:00:00:00:00:16
1188	106.017919	00:00:00_00:00:…	00:00:00_00:00:16	ARP	64	10.1.1.18 is at 00:00:00:00:00:12
1189	106.018132		00:00:00_00:00:12 (00:00:…	802.11	14	Acknowledgement, Flags=........
1190	106.018441	00:00:00_00:00:…	00:00:00_00:00:02	ARP	64	10.1.1.22 is at 00:00:00:00:00:16
1191	106.019448	10.1.1.22	10.1.1.18	AODV	84	Route Reply, D: 10.1.1.8, O: 10.1.1.18 Hcnt=1 DSN=0 Lifetime=11200
1192	106.019661		00:00:00_00:00:16 (00:00:…	802.11	14	Acknowledgement, Flags=........
1193	106.020468		00:00:00_00:00:19 (00:00:…	802.11	14	Acknowledgement, Flags=........
1194	106.020932	10.1.1.22	10.1.1.18	AODV	84	Route Reply, D: 10.1.1.8, O: 10.1.1.18 Hcnt=1 DSN=0 Lifetime=11200
1195	106.021145		00:00:00_00:00:16 (00:00:…	802.11	14	Acknowledgement, Flags=........
1196	106.022761		00:00:00_00:00:02 (00:00:…	802.11	14	Acknowledgement, Flags=........
1197	106.023185	10.1.1.22	10.1.1.18	AODV	84	Route Reply, D: 10.1.1.8, O: 10.1.1.18 Hcnt=2 DSN=0 Lifetime=1003

Figure 3 – Example display of AODV messaging in Wireshark

From Wireshark, these files were exported as JSON files, which can be loaded into a text editor or an IDE to be viewed and analyzed. The tool used in this project was Visual Studio. From here, the key message components of the AODV messages could be located.



```
  "ip.host": "10.1.1.11"
},
"udp": {
  "udp.srcport": "654",
  "udp.dstport": "654",
  "udp.port": "654",
  "udp.port": "654",
  "udp.length": "28",
  "udp.checksum": "0x0000",
  "udp.checksum_tree": {
    "udp.checksum.status": "3"
  },
  "udp.stream": "64",
  "Timestamps": {
    "udp.time_relative": "0.000000000",
    "udp.time_delta": "0.000000000"
  },
  "udp.payload": "02:00:00:07:0a:01:01:02:00:00:00:00:0a:01:01:0c:00:00:22:0f"
},
"aodv": {
  "aodv.type": "2",
  "aodv.flags": "0",
  "aodv.flags_tree": {
    "aodv.flags.rrep_repair": "0",
    "aodv.flags.rrep_ack": "0"
  },
  "aodv.prefix_sz": "0",
  "aodv.hopcount": "7",
  "aodv.dest_ip": "10.1.1.2",
  "aodv.dest_seqno": "0",
  "aodv.orig_ip": "10.1.1.12",
  "aodv.lifetime": "8719"
}
```

Figure 4 –Example of AODV messaging in JSON format view in Visual Studio

The next stage was to extract the relevant features of the AODV messaging being sent and received from each node to and from its tier 1 neighbours and convert them into a dataset that could be used for the machine learning process.

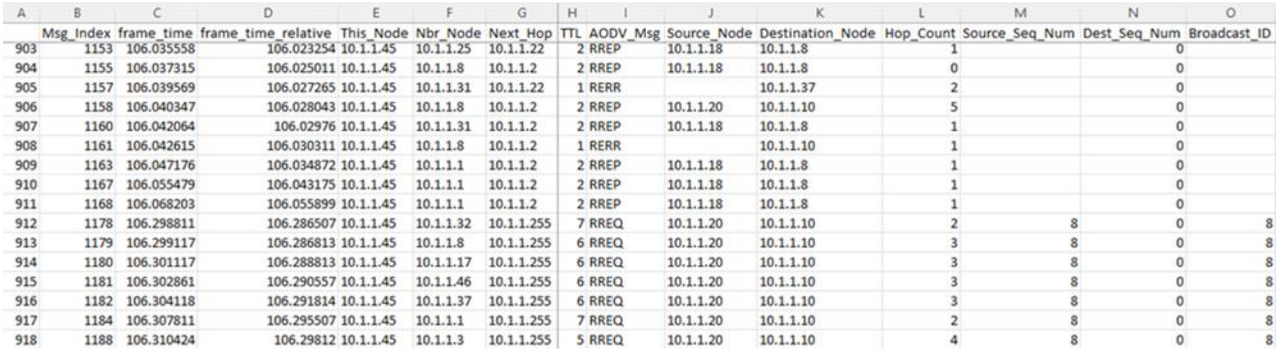
Our team developed a Python script in Jupyter Notebooks to do this. The list of black hole nodes was manually entered into the script. When run, the target variable "Black_Hole_Node" was modified to *True* if the neighbour node was in the list. The rows in the dataset where the black hole node was the subject node were removed from the dataset because the purpose of this process is for a normal node to learn how to detect the behaviour of a black hole node.

This process was split into three stages.

Stage 1

Read in the relevant AODV information from each node (each JSON file) and store them in separate data frames. If it is desired in the future to use another source of AODV messaging as the input into the dataset creation process, only stage 1 needs to be modified to get the relevant message features from the other source. stages 2 and 3 can be left unchanged.

Looking at the example Stage 1 output in Figure 3, each row refers to each extracted message marked by a frame time. It contains information on the Subject Node, Neighbour Node, the AODV Message type and many other relevant features. Columns B to O are the data frames for each node produced by stage 1.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Msg_Index	frame_time	frame_time_relative	This_Node	Nbr_Node	Next_Hop	TTL	AODV_Msg	Source_Node	Destination_Node	Hop_Count	Source_Seq_Num	Dest_Seq_Num	Broadcast_ID
903	1153	106.035558	106.023254	10.1.1.45	10.1.1.25	10.1.1.22	2	RREP	10.1.1.18	10.1.1.8	1		0	
904	1155	106.037315	106.025011	10.1.1.45	10.1.1.8	10.1.1.2	2	RREP	10.1.1.18	10.1.1.8	0		0	
905	1157	106.039569	106.027265	10.1.1.45	10.1.1.31	10.1.1.22	1	RERR		10.1.1.37	2		0	
906	1158	106.040347	106.028043	10.1.1.45	10.1.1.8	10.1.1.2	2	RREP	10.1.1.20	10.1.1.10	5		0	
907	1160	106.042064	106.02976	10.1.1.45	10.1.1.31	10.1.1.2	2	RREP	10.1.1.18	10.1.1.8	1		0	
908	1161	106.042615	106.030311	10.1.1.45	10.1.1.8	10.1.1.2	1	RERR		10.1.1.10	1		0	
909	1163	106.047176	106.034872	10.1.1.45	10.1.1.1	10.1.1.2	2	RREP	10.1.1.18	10.1.1.8	1		0	
910	1167	106.055479	106.043175	10.1.1.45	10.1.1.1	10.1.1.2	2	RREP	10.1.1.18	10.1.1.8	1		0	
911	1168	106.068203	106.055899	10.1.1.45	10.1.1.1	10.1.1.2	2	RREP	10.1.1.18	10.1.1.8	1		0	
912	1178	106.298811	106.286507	10.1.1.45	10.1.1.32	10.1.1.255	7	RREQ	10.1.1.20	10.1.1.10	2	8	0	8
913	1179	106.299117	106.286813	10.1.1.45	10.1.1.8	10.1.1.255	6	RREQ	10.1.1.20	10.1.1.10	3	8	0	8
914	1180	106.301117	106.288813	10.1.1.45	10.1.1.17	10.1.1.255	6	RREQ	10.1.1.20	10.1.1.10	3	8	0	8
915	1181	106.302861	106.290557	10.1.1.45	10.1.1.46	10.1.1.255	6	RREQ	10.1.1.20	10.1.1.10	3	8	0	8
916	1182	106.304118	106.291814	10.1.1.45	10.1.1.37	10.1.1.255	6	RREQ	10.1.1.20	10.1.1.10	3	8	0	8
917	1184	106.307811	106.295507	10.1.1.45	10.1.1.1	10.1.1.255	7	RREQ	10.1.1.20	10.1.1.10	2	8	0	8
918	1188	106.310424	106.29812	10.1.1.45	10.1.1.3	10.1.1.255	5	RREQ	10.1.1.20	10.1.1.10	4	8	0	8

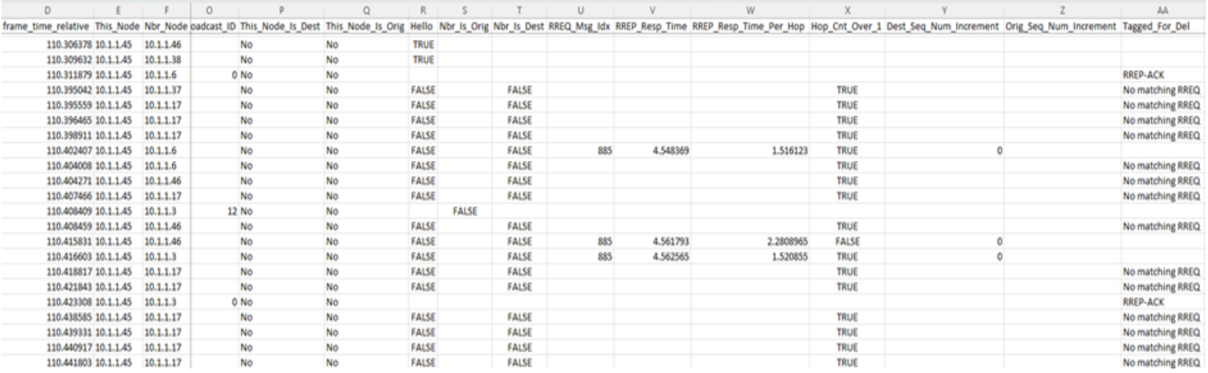
Figure 5 – Example Output of a Data Frame after Stage 1

Stage 2

Some processing is made for each AODV message, creating new features in the data frame: columns P to AA in the Stage_2.csv files. Many of these are Boolean values of if the message is a certain type or is addressed to or originates from the neighbour node.

Suppose the message is an RREP message responding to an RREQ message. In that case, the RREQ message index is found in Column U and the response time between the RREQ and the RREP is noted in columns V & W. If it is an RREP, the destination sequence number increment is also determined in Column Y.

As currently there are some errors with the NS-3 outputs in emulating the AODV protocol, some rows were deleted from the data frames including, for now, RREP-ACK messages and RREP messages from which no corresponding RREQ message could be found.

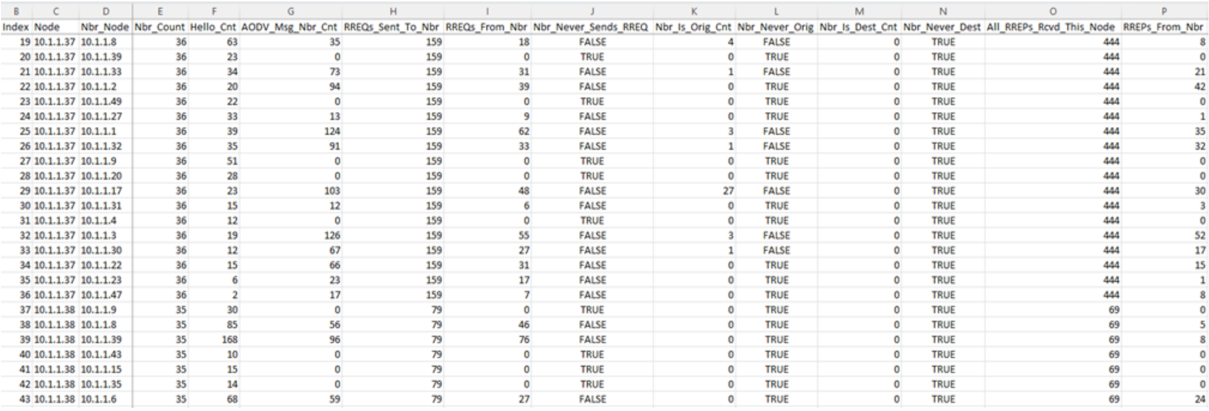


D	E	F	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
frame_time_relative	This_Node	Nbr_Node	oadcast_ID	This_Node_Is_Dest	This_Node_Is_Orig	Hello	Nbr_Is_Orig	Nbr_Is_Dest	RREQ_Msg_Idx	RREP_Resp_Time	RREP_Resp_Time_Per_Hop	Hop_Cnt_Over_1	Dest_Seq_Num_Increment	Orig_Seq_Num_Increment	Tagged_For_Del
110.306378	10.1.1.45	10.1.1.46		No	No	TRUE									
110.309632	10.1.1.45	10.1.1.38		No	No	TRUE									
110.311879	10.1.1.45	10.1.1.6	0	No	No										RREP-ACK
110.395042	10.1.1.45	10.1.1.37		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.395559	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.396465	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.398911	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.402407	10.1.1.45	10.1.1.6		No	No	FALSE		FALSE	885	4.548369	1.516123	TRUE	0		
110.404008	10.1.1.45	10.1.1.6		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.404271	10.1.1.45	10.1.1.46		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.407466	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.408409	10.1.1.45	10.1.1.3	12	No	No		FALSE								
110.408459	10.1.1.45	10.1.1.46		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.415831	10.1.1.45	10.1.1.46		No	No	FALSE		FALSE	885	4.561793	2.2808965	FALSE	0		
110.416603	10.1.1.45	10.1.1.3		No	No	FALSE		FALSE	885	4.562565	1.520855	TRUE	0		
110.418817	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.421843	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.423308	10.1.1.45	10.1.1.3	0	No	No										RREP-ACK
110.438585	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.439331	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.440917	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ
110.441803	10.1.1.45	10.1.1.17		No	No	FALSE		FALSE				TRUE			No matching RREQ

Figure 6 – Example Output of Additional Features of a Data Frame added after Stage 2

Stage 3

This stage will convert the data frames from stage 2 into datasets for each Subject Node and then merge them all into one combined dataset. Every row represents features between the subject node and each of its first-tier neighbouring nodes. Each subject-to-neighbour node relation is only ever one row. The features are mainly counters, percentages as well as Boolean values. All the features are described in the excel file Dataset Features.xls which is included with the deliverables.



B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Index	Node	Nbr_Node	Nbr_Count	Hello_Cnt	AODV_Msg_Nbr_Cnt	RREQs_Sent_To_Nbr	RREQs_From_Nbr	Nbr_Never_Sends_RREQ	Nbr_Is_Orig_Cnt	Nbr_Never_Orig	Nbr_Is_Dest_Cnt	Nbr_Never_Dest	All_RREPs_Rcvd_This_Node	RREPs_From_Nbr
19	10.1.1.37	10.1.1.8	36	63	35	159	18	FALSE	4	FALSE	0	TRUE	444	8
20	10.1.1.37	10.1.1.39	36	23	0	159	0	TRUE	0	TRUE	0	TRUE	444	0
21	10.1.1.37	10.1.1.33	36	34	73	159	31	FALSE	1	FALSE	0	TRUE	444	21
22	10.1.1.37	10.1.1.2	36	20	94	159	39	FALSE	0	TRUE	0	TRUE	444	42
23	10.1.1.37	10.1.1.49	36	22	0	159	0	TRUE	0	TRUE	0	TRUE	444	0
24	10.1.1.37	10.1.1.27	36	33	13	159	9	FALSE	0	TRUE	0	TRUE	444	1
25	10.1.1.37	10.1.1.1	36	39	124	159	62	FALSE	3	FALSE	0	TRUE	444	35
26	10.1.1.37	10.1.1.32	36	35	91	159	33	FALSE	1	FALSE	0	TRUE	444	32
27	10.1.1.37	10.1.1.9	36	51	0	159	0	TRUE	0	TRUE	0	TRUE	444	0
28	10.1.1.37	10.1.1.20	36	28	0	159	0	TRUE	0	TRUE	0	TRUE	444	0
29	10.1.1.37	10.1.1.17	36	23	103	159	48	FALSE	27	FALSE	0	TRUE	444	30
30	10.1.1.37	10.1.1.31	36	15	12	159	6	FALSE	0	TRUE	0	TRUE	444	3
31	10.1.1.37	10.1.1.4	36	12	0	159	0	TRUE	0	TRUE	0	TRUE	444	0
32	10.1.1.37	10.1.1.3	36	19	126	159	55	FALSE	3	FALSE	0	TRUE	444	52
33	10.1.1.37	10.1.1.30	36	12	67	159	27	FALSE	1	FALSE	0	TRUE	444	17
34	10.1.1.37	10.1.1.22	36	15	66	159	31	FALSE	0	TRUE	0	TRUE	444	15
35	10.1.1.37	10.1.1.23	36	6	23	159	17	FALSE	0	TRUE	0	TRUE	444	1
36	10.1.1.37	10.1.1.47	36	2	17	159	7	FALSE	0	TRUE	0	TRUE	444	8
37	10.1.1.38	10.1.1.9	35	30	0	79	0	TRUE	0	TRUE	0	TRUE	69	0
38	10.1.1.38	10.1.1.8	35	85	56	79	46	FALSE	0	TRUE	0	TRUE	69	5
39	10.1.1.38	10.1.1.39	35	168	96	79	76	FALSE	0	TRUE	0	TRUE	69	8
40	10.1.1.38	10.1.1.43	35	10	0	79	0	TRUE	0	TRUE	0	TRUE	69	0
41	10.1.1.38	10.1.1.15	35	15	0	79	0	TRUE	0	TRUE	0	TRUE	69	0
42	10.1.1.38	10.1.1.35	35	14	0	79	0	TRUE	0	TRUE	0	TRUE	69	0
43	10.1.1.38	10.1.1.6	35	68	59	79	27	FALSE	0	TRUE	0	TRUE	69	24

Figure 7 – Example Output of the Dataset produced after Stage 3

Black Hole Node Data Alteration

It must be stressed that black hole nodes simulations were not able to be implemented in the NS-3 simulator within the timeframe of this project, so the values in the datasets have been manually modified for the rows where the target variable "Black_Hole_Node" is set to True. Many of the final features' values have been altered so that the black hole's behaviour is considered "typical" or "dumb", and its characteristics are extremely obvious. This would be the typical behaviour of a black hole node that is not expecting the network nodes to have any intelligence in detecting and mitigating malicious nodes, which is true for most real black hole attacks.

As an example of the data that was altered for black hole nodes, the variable RREQs_From_Nbr was always set to 0 and its corresponding flag feature Nbr_Never_Sends_RREQ was always set to True. This is the behaviour that is expected from a black hole as it would always be responding to an RREQ (Route Request) with an RREP (Route Reply) claiming to have the shortest route to the destination, so it would never broadcast the RREQ further on to its neighbours resulting in a count of 0. Likewise, RREP_Resp_Pct (RREP response percentage) would always be 100% because the Black Hole Node always responds to any RREQ it receives.

For a more detailed explanation of what values were manually altered, please refer to the end of the Dataset Creation section of the project closure report. This script is included in the project deliverables GitHub.

Machine Learning Process

A machine learning script was developed using Python and Jupyter Notebooks to use the datasets created by the conversion script as input. The ML script would be used to develop models to detect which nodes within the AODV network were malicious black hole nodes. An example training and test csv files have been attached in the deliverable package to demonstrate the machine learning script is working. Again, it must be highlighted that many AODV characteristics have not been accurately implemented into the datasets.

The training and test datasets were loaded into the script and some cleaning of the datasets was performed. The Boolean values are converted to integers and any rows with missing values (mainly due to neighbours not carrying any traffic) are removed. The features with the highest correlation to the Black_Hole_Node target variable were selected for the train and test sets that were used in the modelling. After evaluation, a final set of 10 features was selected to be used in the modelling.

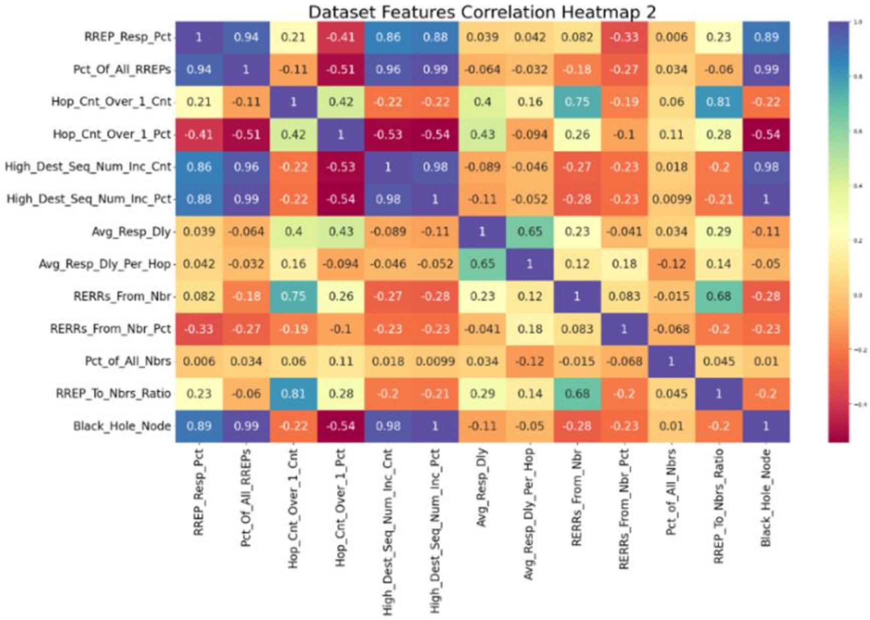


Figure 8 – Example output of the correlation heatmap demonstrating the relationship between the features and the target variable (bottom row)

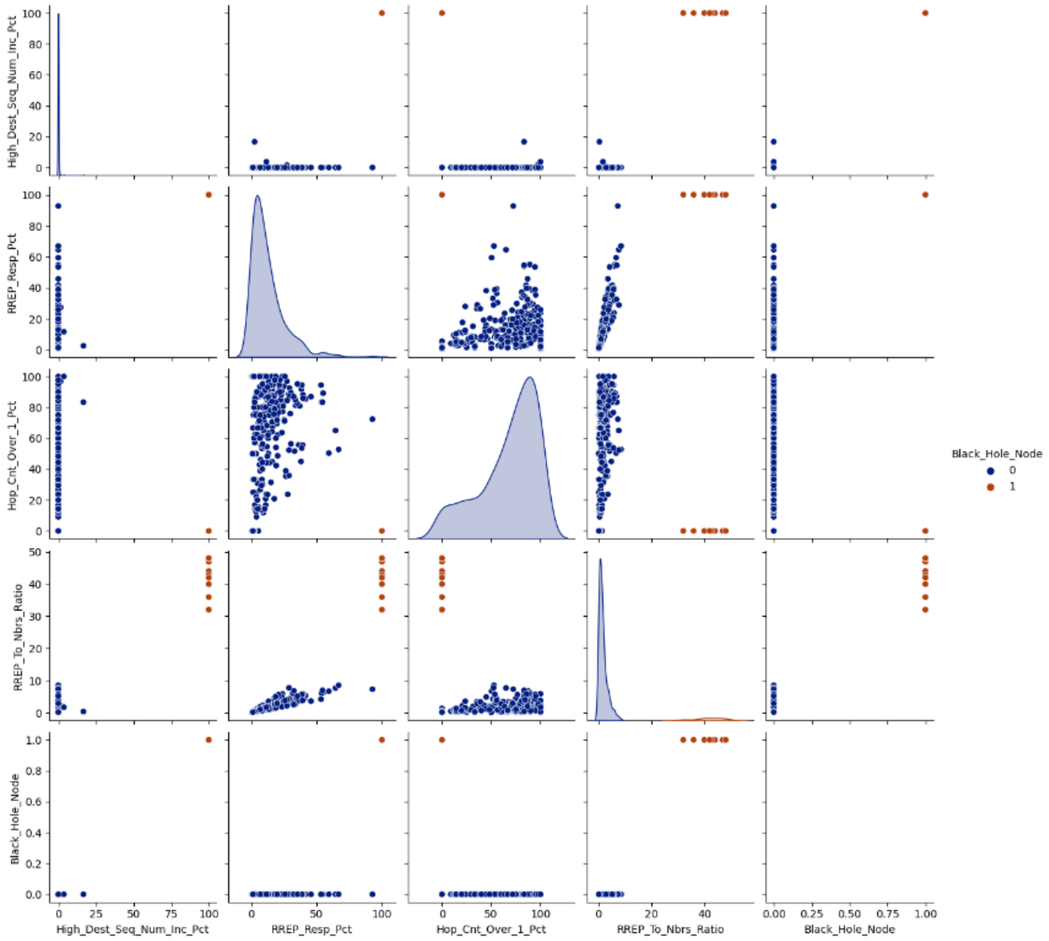


Figure 9 – Example output of pair plots showing the relationships between variables. (Black hole node is brown, nrmal node is blue)

Two models were used to detect the black hole nodes. Model 1 is a Random Forest Classifier and model 2 is a Support Vector Machine (SVM) classifier. A grid search was carried out on both models to determine the optimum hyperparameter settings. Standardization was also applied to the data for the SVM so that all features carry an equal weight in the decision process. The models were trained and were then applied to the test set to determine their accuracy. In both models, their accuracy is 100%.

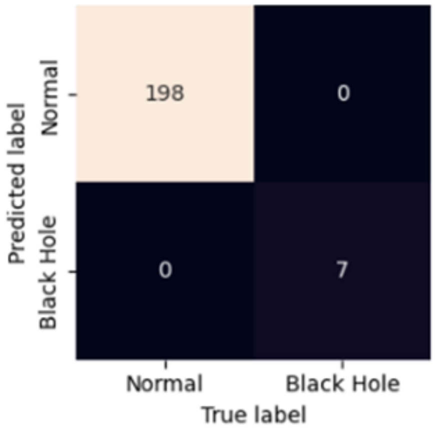


Figure 10 – Example of the Confusion Matrix from the SVM Model Classifying Black Hole Nodes

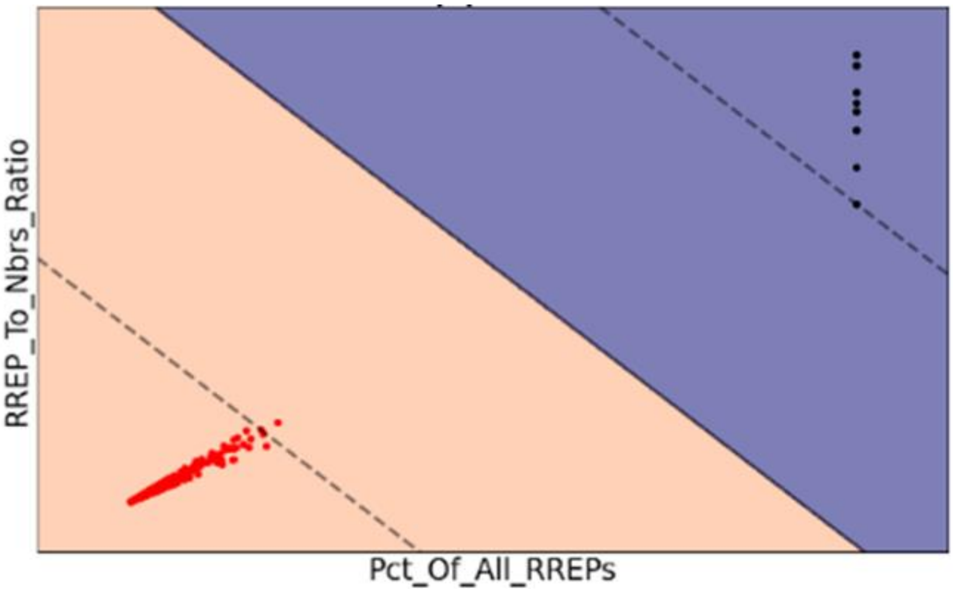


Figure 11 – Example of SVM Visualization of support vectors. There is a large separation between normal nodes (red) and black hole nodes (black)

As mentioned, these models are proof of concept, and further work needs to be done once the NS-3 simulator simulates AODV networks with blackholes accurately and generates complete message logs – this is covered further in the Future Research seciton. The machine learning script is available online via GitHub.

Trained Model Files

Two models were trained to detect black hole nodes operating in an AODV network.

1. A random forest classifier
2. A support vector machine classifier

Both of these models have been 'pickled' as binary files and have been included with the deliverables on GitHub.

Resources Report

The scope of this project did not require many resources to be used. None of the allocated budget was used with the total cost of consumables **$0.00**. There were no hardware requirements. All the software used to develop this project was run on our personal computers, and freely available.

Resources Used for NS3 Simulation	Resources Used for Data Analysis and Machine Learning
Network Simulator 3 (NS3) – NS 3.37	Python 3.10
Host OS WSL2 on Windows 11	Jupyter Notebooks
Linux OS Ubuntu 20.04.5	Pandas
Python 3.8.10	Scikit-Learn
	Matplotlib
	Seaborn

Outstanding Issues

1. NS-3 Network Simulations

Current observed issues in the network simulation output accurately simulating AODV network and black hole node behaviour:
After running the NS-3 simulations. A detailed examination was carried out to determine if the output conforms to AODV protocol behaviour. Unfortunately, numerous issues were found with the simulations of AODV network behaviour. Most of these were issues with the messaging and protocol behaviour logged not containing all the data needed to verify the simulation was strictly following AODV protocol. For the purposes of the project and Model training, it has been assumed that the protocol was, in fact, correctly simulated and the issue is one of logging rather than simulation. This seems justified since the AODV protocol scripts used in the simulation is part of the NS3 code base.

2. NS-3 Black Hole Node Simulations

Another important issue was the accurate simulation of black hole nodes. Unfortunately, the duration of the project did not permit enough time to correct these issues. As mentioned, when split by node, the output message logs available using the PCAP trace output did not give a complete set of messages that must have been passed. In addition, we intended the script to randomly allocate malicious node ids to prevent trained models from picking up on specific node ids as being malicious. However, this requires the node attribute for malicious behaviour to be logged in order to be able to label transactions for training correctly. This proved to be frustratingly difficult. To work around this in development a toy 4 node example black hole attach simulation was used to get blackhole message data. This example data was used as a template and the output from a standard AODV network simulation with 50 nodes was manually altered to show blackhole behaviour in order to be able to complete the workflow for the data processing and model training stages. While ML models were able to be trained with this data, these models still need to be validated against simulation output directly from a working blackhole attack simulation.

3. Data Conversion, Machine Learning Processes and Scripts

There are currently no known issues with these scripts. Once the NS-3 network simulator is producing accurate simulation trace files for each node, the logic of the data conversion script and the machine learning script must be re-verified and thoroughly tested to ensure they are correctly processing the pcap files. Undoubtedly there will be several improvements or features that can be discovered and implemented in these scripts as they are used and tested more.

Despite these issues, we were able to engineer a solution for the project to progress and successfully complete the machine learning stage. A complete list of known issues is detailed in the *Outstanding Issues* section of the Project Closure Report.

Project Risks

Risk	Risk Description	Likelihood	Impact	Status	Mitigation
Personnel Issues	It is possible that any project member could become sick or may have to leave the project due to personal issues.	Moderate	Moderate	Closed - Treated	One member became sick for a few weeks. Overlapping responsibilities with other members. Two members were simultaneously working on the NS-3 simulations.
Technical Difficulties	Difficulties with configuring the NS-3 environment and building and running the scripts.	Moderate	Severe	Closed - Treated	Several difficulties with configuring the NS-3 environment and building and running the scripts. Two different NS-3 environments were set up on two different machines. This enabled the team to progress further with the project. Output datasets were altered to accurately simulate black hole node behaviour.
Hardware Requirements	Personal Computers may not possess sufficient processing power.	Low	Low	Closed - Mitigated	AWS remote environment will be utilized for the NS-3 simulation.
Machine Learning Software Configuration	Issues with running Python scripts or other required software on personal PCs.	Low	Low	Closed - Avoided	Using online platforms such as Google Colab for the ML training.

Hardware Issues	A PC used on the project may be damaged or stolen.	Moderate	Moderate	Closed - Mitigated	1. Files will be checked out of team GitHub. 2. Word documents will be edited on the team's online file repository. 3. Other files should be stored on the team's online file repository.
Unforeseen Project Delays	Project delays if there are unresolved issues that impede progress along the way. This could result in additional costs if the project were to be delayed.	Moderate	Severe	Closed - Mitigated	The project scope will be clearly defined before beginning and ensure that the entire project can be completed within the designated timeframe. By doing so, we aim to prevent potential delays and minimize any associated additional costs that could arise due to project overruns.
Unrealistic Expectations	Unclear or Unrealistic Expectations can arise when there is a lack of mutual understanding among all parties involved in the project. Unrealistic expectations may lead to unattainable project goals or lower quality standards.	Moderate	Moderate	Closed - Mitigated	A careful planning process, including the estimation of costs and time required to complete milestones. Additionally, effective communication will be established to ensure that the entire team has a clear understanding of the project's objectives and expectations.
Not Meeting Deadlines	Missing Deadlines and Deliverables.	Low	Severe	Closed - Mitigated	Prioritise tasks by allocating more resources to those that are crucial to the project and giving them the highest priority. Conversely, low-priority tasks will receive fewer resources. Identification of dependencies and completion of tasks synchronously whenever possible. Implementing these

					measures will ensure that all critical deadlines and deliverables are met on time.
Poor Communication	Poor communication among team members and stakeholders.	Moderate	Severe	Closed - Mitigated	Stakeholders will be informed of the most effective communication channels. Team members will meet regularly and provide weekly updates to the project manager on the progress of their assigned tasks. Sponsors will receive regular progress reports. This will ensure that all team members are well-informed and that there is no miscommunication. By implementing this strategy, communication will improve, and this will facilitate a better understanding of the project's progress among all stakeholders.

Lessons Learned

The importance of a good risk management plan: Having a good risk management plan is crucial for projects such as this. With a relatively short time frame for this project and many challenging technical tasks, one serious technical hurdle could likely cause lengthy delays and ensure that not all of the project's scope can be completed. This was the case with difficulties encountered with running the NS-3 simulations. We mitigated this to a certain extent by having two team members work on these simulations. One member successfully generated simulation files that could be edited and then used to complete the data conversion and the machine learning processes. Achieving these objectives greatly helped to make this project a success.

Ethical Issues: All software and scripts used were open source and all the data used in this project were generated from scripts, so it is nobody's real or personal data. As such, there are no ethical issues with the actual source or privacy of the data. One ethical lesson learned with the use of the data is to always be honest and upfront with its integrity. As has been mentioned due to simulation issues, in the datasets we modified the values of several features for black hole node rows to accurately represent the behaviour of a black hole node. This has been done as accurately as possible and we believe the values are very realistic. However, being ethical and honest about any changes we have made is always imperative. We have clarified this to our sponsor and mentor, stating that this was the best option for us to proceed with the project. Of course, once the NS-3 simulations are corrected, the unedited simulation data would be used, and the models' training and testing would be reassessed.

Practical limitations on sharing work: Microsoft Teams and shared documents are very useful for report writing; however, joint programming is more difficult to share effectively remotely. The lack of access to a shared programming environment caused difficulties in sharing ideas about how to deal with difficulties working with the NS3 simulations, and limited other team members, insight into the state of the simulation code.

A shared responsibility model was a good decision: the simulation task was structured in such a way that initial data for a vanilla network was produced early on. This enabled the work on data processing and ML training to proceed in tandem with the simulation work. In the end, the lack of a fully working blackhole simulation of the right size did limit what was achievable with the ML, however, it did not prevent proof of concept work.

Difficulties with setting up working environments should not be underestimated: numerous difficulties were encountered in setting up the environment for NS-3 with Linux as described in the risks mitigated section. The initial difficulties related to non-native Linux machines and being able to install all the required dependencies on these. Running simulations on a dedicated remote VM with a native Linux OS would help to mitigate these difficulties. It would also make work sharing on the simulation easier.

Configuring and running NS-3 simulations: further problems related to the difference in the NS3 versions between the source scripts that we intended to use and the default installations of the latest NS3 version. Significant changes were made to the NS3 simulation environment between version 3.25 and version 3.30 including removing some modules that were used in the source scripts. These changes substantially increased the amount of work involved in creating the simulation scripts for our scenarios.
Lesson learned: the initial effort of making sure that the installed versions of all the OS and software components are synchronized and compatible and that they match the versions of the source scripts, is worthwhile. Open-source software offers flexibility but there is more complexity in configuring and matching versions than might be found in commercial software.

Handover Materials to the Sponsor

With a project of this scope, many separate scripts were developed, and output files created. These materials plus accompanying documentation were handed over to the project sponsor.

These materials include:

- Project Closure Report.
- Literature Review.
- Jupyter Notebooks Script to convert the network trace files of each node into a dataset.
- Jupyter Notebook Script to create machine learning models to detect and identify black hole nodes from the datasets.
- Excel sheet explaining dataset features.
- Tuned Random Forest and Support Vector Machine models to detect and classify black hole nodes.
- Zip file of example node trace files (pcap) and generated datasets that were used for the machine learning phase.
- Modified AODV routing protocol modules
- Modified Manet routing compare script
- Notes on NS3 Version installation and environment setup.

JSON files are not included due to their immense size. They can easily be created by loading pcap files into Wireshark and converting them to json files.

To view all the handover materials, please refer to our GitHub repository.

Recommendations

It has already been well documented that the network simulations were not running as expected according to the AODV protocol. Despite this, the project overall has been a success and it holds much potential that a very accurate solution to detecting and circumventing black hole attacks can be achieved through machine learning techniques. It is recommended that further work be carried out on this project as we feel confident that a good working solution can be achieved. Several recommendations can be made to the sponsor and anyone wishing to continue working on this project. They are explained in the future research section. However, the recommendations with the highest priorities are:

- To ensure the NS-3 simulations are working and modelling black hole nodes accurately.
- With accurate simulation trace files, thoroughly test the data conversion and machine learning scripts, to ensure that the scripts are running as expected and can produce accurate results.
- Obtain larger datasets comprising several simulations with different network configurations should be combined to train the models to detect black hole nodes in various conditions.

Future Research

Many exciting extensions to this work could be done to improve the applicability of this project and its effectiveness at nullifying black hole nodes in real life scenarios. This is beyond the scope of a one semester project. Several essential steps include:

- Improve the NS-3 simulations so that the simulations with blackhole nodes are behaving as expected according to the AODV protocol.
- Improve the black hole node simulations. For the first stage of simulations, the black hole nodes should be "dumb" where their behaviour is obvious, i.e., they act exactly as described in the Project Closure Report.
- Once these are working accurately, create several datasets with various network configurations, including:
    - Network size
    - Number of nodes,
    - Simulation run time
    - Mobility of nodes
    - Activity of nodes
    - The number of black hole nodes.
- Combine all of the various network configuration datasets into one large training dataset. This larger set will be used to train the models to various configurations of networks, so that it may more accurately identify black hole nodes in certain network conditions. Blackhole nodes should not be the subject node in the dataset, as it is desired to train normal nodes to detect and identify black hole nodes.
- The test dataset should also be large and consist of a wide variety of network conditions.
- After the models have been trained, a new network simulation can easily be configured and run and used to detect the model's accuracy at detecting the black hole nodes.

- After the final models have been trained, they can be deployed into the normal nodes within the network. These normal nodes will monitor their first-tier neighbours' behaviour for a certain period and should be able to detect if a first-tier neighbour is a black hole node. If so, the subject node should be trained to find alternative routes to the desired destination, effectively avoiding and isolating the black hole node.
- Further work from the previous step could be to tune how much activity data is required for the subject node to detect the black hole nodes accurately. The idea is to minimize the amount of time to detect the black hole node so that disruption to network traffic is minimized.
- A further step could be to use the NS3-AI module to deploy ML models in the simulated nodes in MANET, to mitigate the effects of a black hole attack. Once the AI detects the malicious nodes, the return value would be used to update the routing table to prevent further disruption by avoiding the affected nodes.
- The effectiveness of the ML models in preventing network performance degradation in the event of a black hole attack could be determined in comparisons between simulations with and without deployed models.
- Total network traffic could be monitored for both:
    1. Normal nodes
    2. All nodes made "smart" by incorporating the ML algorithms to detect and bypass black hole nodes

The difference in total network traffic degradation could be monitored and compared from the two scenarios.

- Progressively make the black hole nodes "smarter" so that their behaviour is not consistently so obvious and repeat the network simulations to test the models' accuracy detection rate.
- Write and publish a paper about the project and the methods developed to detect and combat black hole attacks in MANET networks.

Sponsor Signoff

Ask the sponsor to Complete the "proforma" and ask for a formal signoff and include it in the report.

References

[1] *MANET Routing Protocols using ns3*. (n.d.). Retrieved April 20, 2023, from https://www.nsnam.com/2019/05/comparison-of-adhoc-routing-protocols.html

[2] *Mohit P. Tahiliani: [ns-3] Blackhole Attack Simulation in ns-3*. (n.d.). Retrieved April 20, 2023, from http://mohittahiliani.blogspot.com/2014/12/ns-3-blackhole-attack-simulation-in-ns-3.html

[3] Kumar, T. S. P. (2014, February 25). *Adding a malicious node in NS2 in AODV Protocol*. NSNAM.Com. https://www.nsnam.com/2014/02/adding-malicious-node-in-ns2-in-aodv.html

[4] *ns-3: src/aodv/model/aodv-routing-protocol.cc File Reference*. (n.d.). Retrieved April 20, 2023, from https://www.nsnam.org/docs/release/3.28/doxygen/aodv-routing-protocol_8cc.html