ARMY RESEARCH LABORATORY

From Malicious Eyes: A Method for Concise Representation of Ad-Hoc Networks and Efficient Attack Survivability Analysis

by Jaime C. Acosta

ARL-TR-6035 **July 2012**

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
White Sands Missile Range, NM 88002-5501

ARL-TR-6035 **July 2012**

From Malicious Eyes: A Method for Concise Representation of Ad-Hoc Networks and Efficient Attack Survivability Analysis

Jaime C. Acosta
Survivability/Lethality Analysis Directorate, ARL

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2012	Final	October 2011–September 2012

4. TITLE AND SUBTITLE
From Malicious Eyes: A Method for Concise Representation of Ad-Hoc Networks and Efficient Attack Survivability Analysis

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jaime C. Acosta

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-SLE-I
White Sands Missile Range, NM 88002-5501

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-6035

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Ad-hoc networks are undergoing widespread use because they are able to provide capabilities that are infeasible with traditional infrastructure networks such as adaptive topologies. With these capabilities, however, security risks and resource limitations are introduced. Much work has focused on the development of security strengthening mechanisms such as secure routing protocols, traffic encryption, distributed intrusion detection, and others. Many times, these methods are not feasible due to limitations of available processing and power. In addition, it has long been known that security can never be guaranteed.
Alongside security analysis, survivability analysis focuses on the ability of network entities to function even during attacks. While previous methods attempt to measure system tolerance as a whole, for example, average throughput over several simulations, still missing are methods that are able to predict low-level attack impacts. Analysis of these low-level impacts enables analysts to tune and redesign networks to optimize survivability. In this report, a dataset is collected and formatted into a novel network representation. This representation is then used to build a classifier that accurately predicts link loss due to spoofing and data forwarding attacks.

15. SUBJECT TERMS
MANET, attack survivability analysis, machine learning, network representation, ad-hoc routing attacks, risk analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	38	Jaime C. Acosta
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER (*Include area code*) (575) 678-8115

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures iv

List of Tables iv

Acknowledgments v

Summary vii

1. Introduction 1

2. Background 3

3. Preliminary Analysis 5

3.1 Emulation Environment 5

3.2 Attack Development 5

3.3 Observations 6

4. Data Collection 6

4.1 Scenario Generation 6

4.2 Log Data Collection 10

5. Network Representation 11

6. Evaluation 13

7. Results 14

8. Conclusions and Future Work 22

9. References 23

List of Symbols, Abbreviations, and Acronyms 26

Distribution List 27

List of Figures

Figure 1. A sample ad-hoc network. Nodes in the network are routers (blue cylinders are legitimate while the red cylinder is compromised) that are connected (green lines) to other nodes as decided by the routing protocol. 1
Figure 2. Spoofing attack pseudo code. 6
Figure 3. *configGen* script pseudo code 7
Figure 4. Chain topology. 8
Figure 5. Connected grid topology. 8
Figure 6. Cycle topology. 8
Figure 7. Star topology. 8
Figure 8. Tree topology. 9
Figure 9. Two-centroid Topology. 9
Figure 10. Wheel topology. 9
Figure 11. Attacker log tshark flags. 10
Figure 12. Non-Attacker log mgen flags. 11
Figure 13. Representation by hops. Hop counts are labeled in parentheses and dotted lines indicate traffic. 12
Figure 14. OLSR Forwarding survivability model. 16
Figure 15. OLSR Spoofing survivability model part 1. 17
Figure 16. OLSR Spoofing survivability model part 2. 18
Figure 17. OLSR Spoofing survivability model part 3. 19
Figure 18. OSPFv3MDR Forwarding survivability model. 20
Figure 19. OSPFv3MDR Spoofing survivability model part 1. 20
Figure 20. OSPFv3MDR Spoofing survivability model part 2. 21

List of Tables

Table 1. Traffic Data flows between nodes. Column 1 indicates the source node while columns 2 and 3 indicate the nodes being sent TCP and UDP packets respectively. 10
Table 2. Network representation parameters. 12
Table 3. Percentage of flows that conflicted per protocol. 15
Table 4. Weighted averages for classification of *duringLinkLost* with OLSR. 15
Table 5. Weighted averages for classification of *duringLinkLost* with OSPFv3MDR. 15

Acknowledgments

I would like to thank Ernie Martinez and David Nevarez for their help with the attacks. I would also like to thank Naval Research Laboratory and the developer’s of the common open research emulator (CORE) for their feedback during this research.

INTENTIONALLY LEFT BLANK.

Summary

As part of its mission, the U.S. Army Research Laboratory/Survivability Analysis Directorate (ARL/SLAD) evaluates government technologies and provides guidance to ensure maximum system security in the areas of tamper protection, reverse engineering protection, and vulnerability detection and protection among others. As part of these efforts, network survivability is a critical element. Field testing is extremely costly and time consuming. As a result, emulation has become an important analysis resource. In order for ARL/SLAD to provide accurate and efficient survivability analysis, state-of-the-art emulation tools must be adopted and expertise in their design and use is critical. More importantly, research and development of methods that improve efficiency, analysis capability, and use of the scientific method is paramount to the ARL/SLAD mission.

In this report, the notion of attack survivability prediction is introduced. The report provides evidence showing that emulation runtime logs and a carefully designed network representation enable analysts to predict communication flows that are affected by network attacks such as spoofing and data forwarding.

Current methods of survivability analysis do not generalize across scenarios, provide low-fidelity results, and lack scientific backing. The following contributions are made in this report.

1. A network representation is presented that captures the data flows and routes of a wireless ad-hoc network throughout a simulation or emulation. The network representation is concise because it is based on the attacker's view of the network and consists of only 22 parameters.

2. The network representation is evaluated using the common open research emulator (CORE) with several scenarios. Experiments using optimized link state routing (OLSR) and OSPFv3MDR were conducted and 10 fold-cross validation shows that link loss due to spoofing and data forwarding attacks can be accurately predicted, above 97 percent true positive rate and 10 percent false positive rate, across seven distinct scenarios.

INTENTIONALLY LEFT BLANK.

1. Introduction

Many wireless technologies rely on centralized infrastructures to provide services such as public Internet access. Infrastructure-less or ad-hoc networks are meant to serve different purposes. These networks enable wireless entities to communicate over long distances without the need for centralized management. Ad-hoc networks are designed to adapt to environmental changes and require low maintenance. Ad-hoc networks have numerous applications spanning many diverse fields; these applications include military field exercises, intelligent transportation, environmental monitoring, and others (*1*). Figure 1 illustrates an example ad-hoc network.

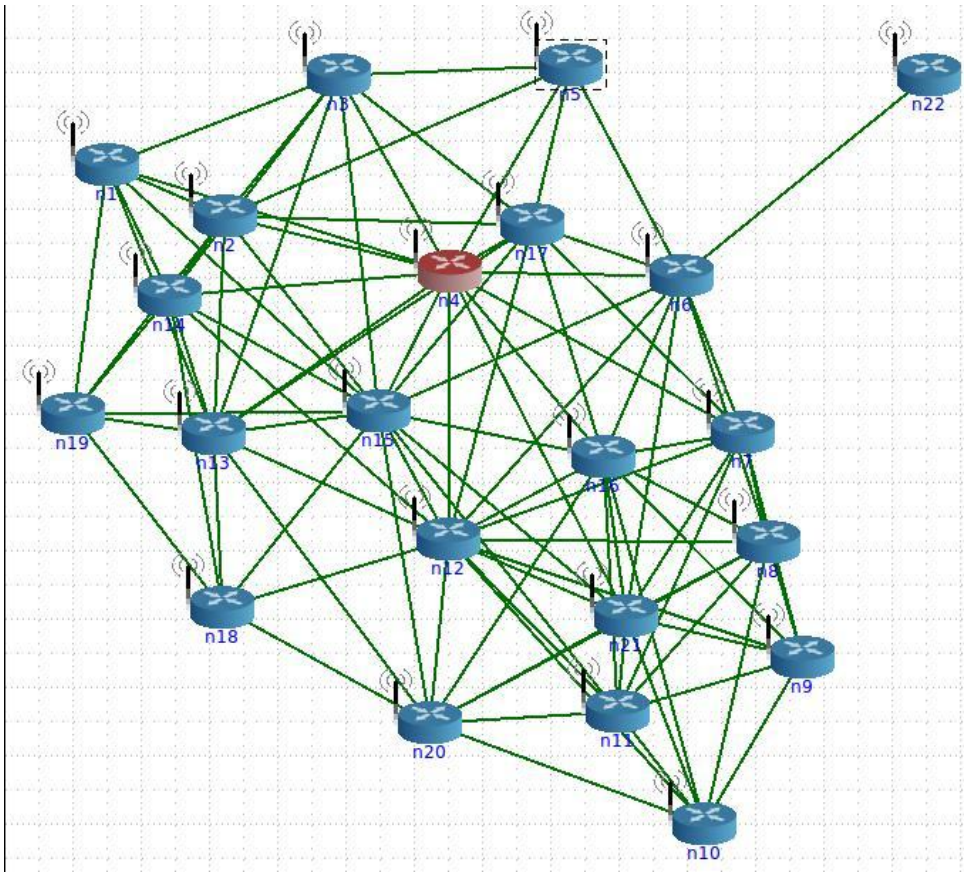

Figure 1. A sample ad-hoc network. Nodes in the network are routers (blue cylinders are legitimate while the red cylinder is compromised) that are connected (green lines) to other nodes as decided by the routing protocol.

Given the increase in the capabilities and the wide use of wireless networks, privacy and security has become a critical research focus. In particular, the benefits of ad-hoc networks introduce additional security vulnerabilities, resource constraints, and performance limitations.

Many have investigated the attacks that are associated with ad-hoc networks. Attacks are classified as passive or active. Passive attacks include eavesdropping, traffic analysis, and monitoring. Active attacks include jamming, spoofing, modification, replaying, and denial of service (*2*).

As a result, countermeasures have been developed, such as distributed intrusion detection, trust management, secure routing protocols, and others. It is well known that ensuring complete security is not feasible. In the field of survivability, research focus lies on improving critical systems' tolerance to incidents.

In the past, several methods have been used to test wireless ad-hoc networks. It can be observed from figure 1 that visual inspection alone is not suitable for analyzing ad-hoc networks as they can become very complex. Field testing, where actual hardware and software are tested in real environments, is non-trivial and costly. Static analysis, such as model checking and proof-based techniques are rigorous, complex, and limited to small non-mobile systems. Current simulation and emulation techniques are useful because they allow analysts to design and test particular scenarios using a combination of software and hardware to draw conclusions from empirical evidence. The problem with current simulation and emulation methods is that results are specific for the scenarios under test. Traditionally, analysts run several instances consisting of different attack and topology parameters. These parameters are mostly chosen at random. This task can become very time consuming, especially when using emulation to provide more accurate results.

Lacking are methods that learn from previous executions in order to predict survivability. In this report, I provide the following contributions.

1. A network representation that captures the data flows and routes of a wireless ad-hoc network throughout a simulation or emulation. The network representation is concise because it is based on the attacker's view of the network.
2. The network representation is evaluated using common open research emulator (CORE) with several scenarios. Experiments using OLSR and OSPFv3MDR were conducted and 10 fold-cross validation shows that link loss due to spoofing and data forwarding attacks can be accurately predicted across seven distinct scenarios.

The report is organized as follows.

1. A background of wireless ad-hoc networks is given.
2. I provide a preliminary analysis that led to this work.
3. The data collection method and the network representation parameters are described.
4. The evaluation follows by presenting the experimental procedure and results.
5. I conclude and offer prospective extensions to this work.

2. Background

Wireless ad-hoc networks are used when communicating entities must be able to adapt in dynamic, long-range environments without the need of a static infrastructure for packet routing decisions. These systems are many times limited in resources, such as electrical and computational power. For these systems to work, the underlying communication relies on efficient, reliable, ad-hoc routing protocols. While there has been much work evaluating the performance of these protocols (*3–5*), security and survivability of these networks has recently become a strong research focus.

Attacks that exist in infrastructure networks may be more difficult to detect in wireless ad-hoc networks. Ad-hoc networks are also susceptible to a broader range of attacks; (*2*) describes such attacks. Wireless ad-hoc networks rely heavily on routing for communication; therefore, there has been much focus on routing protocol security.

Some techniques for analyzing the security of routing protocols are exhaustive (*6*). These techniques attempt to represent a system using mathematics, and then attempt to prove security goals. In these cases, invalid states indicate malicious activity. There are several limitations to techniques. They are unable to cover all conditions, especially in large and mobile systems. Evaluation often requires rigorous analysis of the specification and sometimes conversion to specialized formats. As with any security evaluation technique, exhaustive approaches are prone to false positives in real environments, which may be caused by legitimate system failures.

Recently, there has been an interest in the development of secure protocols, which are designed with security in mind. These protocols contain logic to prevent malicious activity. Common techniques used for ensuring security include cryptographic primitives, (*7–10*), and obfuscation approaches, for example, multipath routing partitions data and sends packets through several (possibly non-optimal) paths (*11, 12*). Secure protocols are not without limitations; these techniques introduce overhead that may not be feasible in some systems due to electrical power and computational constraints. Changing the underlying routing protocol in large legacy systems may be costly.

A non-exhaustive method for evaluating the security of routing protocols is simulation. Using this approach, the security of a network is measured by configuring multiple scenarios with varying conditions, such as topology and routing protocol. During the scenarios, a simulated attack is executed and performance attributes (delay, throughput, goodput, etc.) are measured (*13–15, 6*). Although not exhaustive, well-designed simulations may be a credible source for evaluating security (*16*). Drawbacks of this approach include inaccuracies due to the fact that the network stack and running processes are simulated. To account for this limitation, emulation, which is capable of executing real binaries in actual runtime environments (operating system,

network stack, etc.), is sometimes used for evaluation. In either case, in addition to being non-exhaustive, results from the previous work do not generalize to untested scenarios. Regardless of the evaluation techniques used, due to factors such as field environment, protocol implementation, malicious insiders, user error, etc., unconditional security is not guaranteed.

While security evaluation methods focus on attack prevention and reaction, survivability techniques also take into account tolerance (*17*). Survivability measures how well systems can operate during attacks, intrusions, failures or accidents (*18*). This is useful, for example, when considering networks with critical data. In this case, attacks on nodes in the critical data path would likely have higher impact on the system survivability.

Much research has looked at improving survivability by introducing tolerant routing methods, which may either replace or execute alongside current protocols (*17*). Regarding survivability evaluation, usually either full enumeration is attempted or the monte carlo method is used (*19–21*). These approaches are infeasible with large mobile systems and results do not generalize across to unseen scenarios. Recent methods use machine learning to predict system survivability (*22*). Survivability measurement parameters such as number of critical links and number of surviving paths are averaged over multiple executions. This low-fidelity approach limits further analysis of details that could improve generalization across scenarios and help fine-tune systems for improved survivability.

The method presented in this report uses an attacker-focused representation of ad-hoc networks (network states) that enables accurate prediction of link loss between nodes given real-world attacks. This allows analysts to determine vulnerable network states that are critical during a scenario, and identify alleviations that may result by re-positioning, modifying critical data paths, changing protocols, etc. Network states consist of parameters collected from route dumps and observed traffic flows and were defined based on emulation experimentation.

3. Preliminary Analysis

3.1 Emulation Environment

It was apparent after reviewing the literature that there was a need for a method that could provide survivability prediction for ad-hoc networks. The first step towards developing such a method was to determine how systems, real-world software and hardware, behave when subjected to different attacks. CORE (*23*) was used as the emulation platform for the following reasons:

- Open source: Besides being free, it is also possible to modify and conduct deep analysis of the emulator internals.
- Extensible: Through its plugin architecture, in-house and third party developed components such as EMANE (*24*) for layer 1 and layer 2 emulation and CommEffect (*25*) for emulating real-work communication effects are loadable modules. A developer can also implement custom radio models.
- Maintained: Releases and bug fixes are ongoing. A user's and developer's mailing list allows the community to ask questions.
- Accurate: Network layers 3 and above run in a virtualized Linux environment. Each node runs real binaries and has a real network stack.
- Flexible: Features such as hardware-in-the-loop allow connectivity with real devices. Multiple instances of CORE allow scalability, which is essential for large networks.

3.2 Attack Development

After choosing the emulation platform, the next step was to observe the impact of network attacks on nodes. Out of the box, CORE provides several routing protocols including Quagga's OSPFv3MDR for wireless ad-hoc networks. To broaden the scope of the analysis, NRLOLSR was additionally installed. Two attacks were implemented, namely spoofing and data forwarding, which are well-known in the security community (*2*). These were chosen because they require only basic networking knowledge; they do not require an understanding of underlying algorithms.

The pseudo code for the spoofing attack is provided in figure 2. The spoofing attack takes as input an Internet Protocol (IP) address that will be spoofed. If the OSPFv3MDR protocol is in place, the attack creates a virtual interface and assigns it the IP address of the victim. If OLSR is used, in addition to starting the virtual interface, since multiple interfaces are not supported with NRLOLSR, the attacker broadcasts itself as a host network announcement (HNA) gateway with the IP address of the victim. In the latter case, the routing daemon requires a restart.

```
if olsr daemon is running
        stop olsr daemon
        start virtual interface with $ipToSpoof
        restart olsr daemon with HNA
        sleep $duration
        stop olsr daemon
        stop virtual interface
        restart olsr daemon without HNA
else
        start virtual interface with $ipToSpoof
        sleep $duration
        stop virtual interface
```

Figure 2. Spoofing attack pseudo code.

The data forwarding attack simply drops all data packets that are meant for outside nodes. Control packets are still forwarded. To implement the data forwarding attack the kernel IP forwarding variables (net.ipv4.conf.all.fowarding and net.ipv6.conf.all.fowarding) are set to 0. This process is the same regardless of the protocol used and does not require restarting any processes.

3.3 Observations

Informal testing with several scenarios using transmission control protocol (TCP) and user datagram protocol (UDP) traffic provided limited evidence that during attacks, the communication near the attacker are impacted more often. More specifically, it seemed as though impact was related to distance and flow, based on the attacker's relative location. To investigate this further, I generated an experimentation platform which consists of automated data collection and a network representation focused on the attacker's perspective of the network.

4. Data Collection

Two main components are necessary for the collection of data. Scenario generation is associated with configuration parameters for CORE executions. Log generation is associated with the data collected by individual nodes during each execution.

4.1 Scenario Generation

CORE allows analysts to design and run network scenarios using a graphical interface. Generating topologies is a trivial task that works by dragging and dropping icons into a workspace. Scenario parameters such as node positions, protocols used, and custom processing (such as logging and attacker) scripts are stored in a configuration file. A dataset was generated

by running several scenarios. Each scenario used different parameter values. This was accomplished by implementing a configuration generator, *configGen*. Pseudo code for *configGen* is provided in figure 3.

```
for attackNode in 1…10
        for topology in “chain” “connected_grid” “cycle” “star” “tree” “two-centroid” “wheel”
                for protocol in “OLSR” “OSPFv3MDR”
                        for attack in “forwarding” “spoofing”
                                runScenario($attackNode,$topology,$protocol,$attack)
```

Figure 3. *configGen* script pseudo code

The parameters that were varied with each emulation instance are discussed below

Routing protocol. This is the underlying layer 3 protocol that will be used to communicate data necessary for route maintenance. The dataset contains OLSR and OSPFv3MDR protocols. Both of these protocols are proactive meaning that they continually publish route information, as opposed to reactive protocols, which publish route information when requested. The OLSR implementation is provided by NRL and uses IPv4 addressing. OSPFv3MDR is part of the Quagga suite of protocols. OSPFv3MDR is a modification of OSPF that is optimized for mobile ad-hoc networks. OSPFv3MDR uses IPv6 addressing.

Topology. In attempts to achieve a suitable data distribution, rather than using randomly generated topologies, which are sometimes either too sparse or compressed, 10 nodes were used to populate 7 different static topologies. The topologies were generated by first manually positioning nodes to fulfill the desired connectivity. Next, the position values were hardcoded into the *configGen* script. Figures 4−10 are graphical portrayals of the topologies. The topologies are labeled chain, connected grid, cycle, star, tree, two-centroid, and wheel.

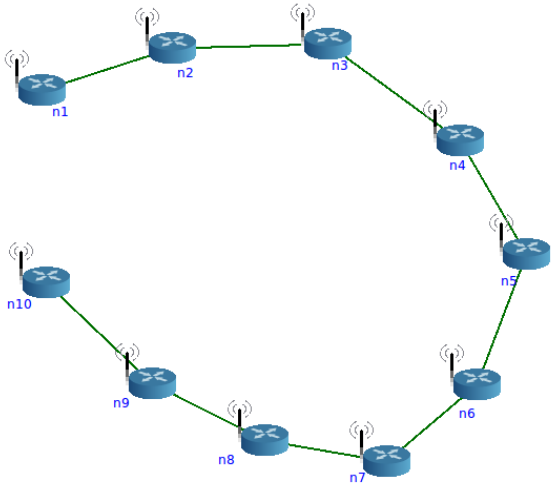

Figure 4. Chain topology.

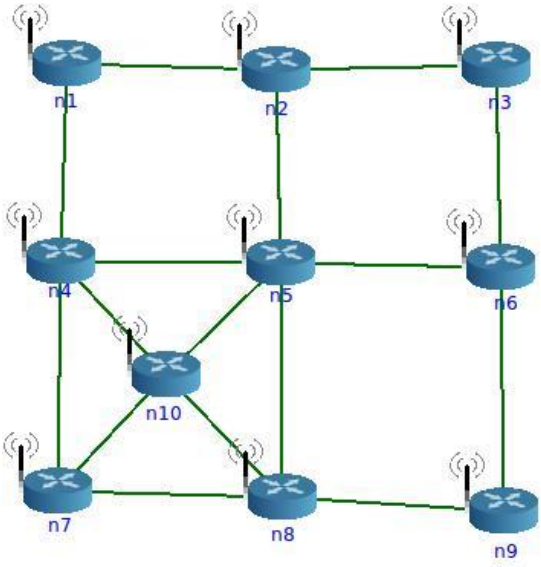

Figure 5. Connected grid topology.

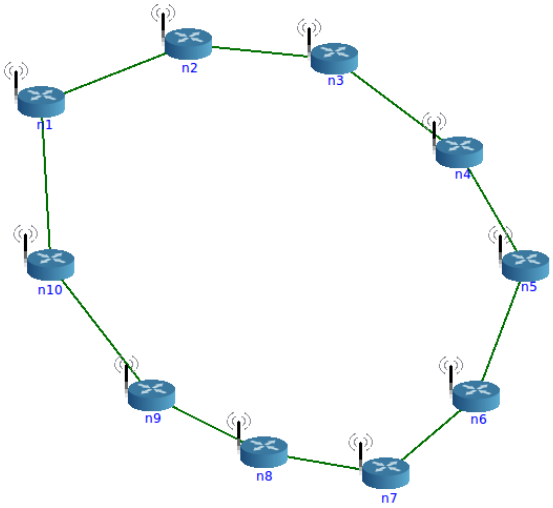

Figure 6. Cycle topology.

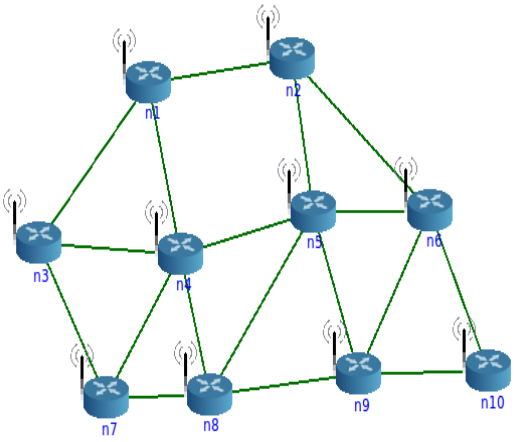

Figure 7. Star topology.

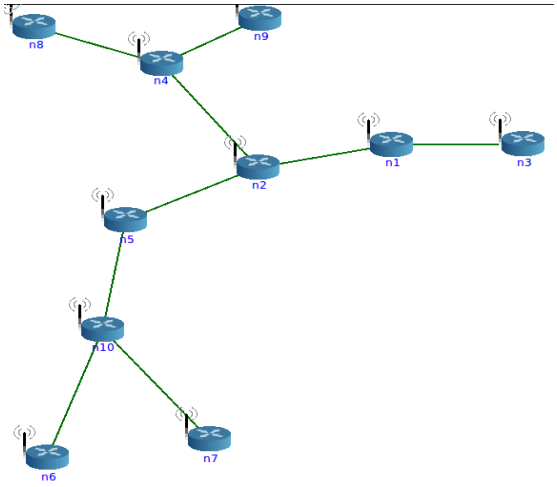

Figure 8. Tree topology.

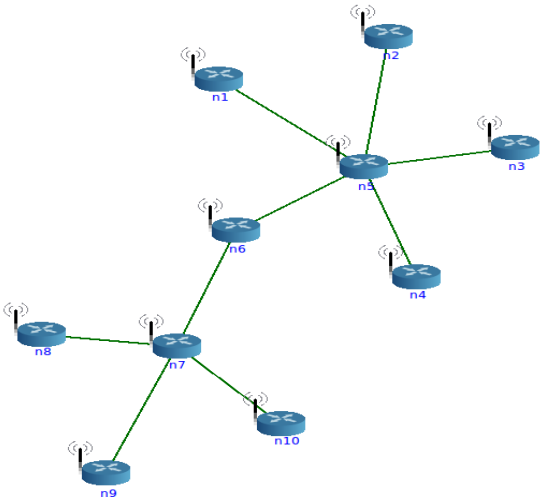

Figure 9. Two-centroid Topology.

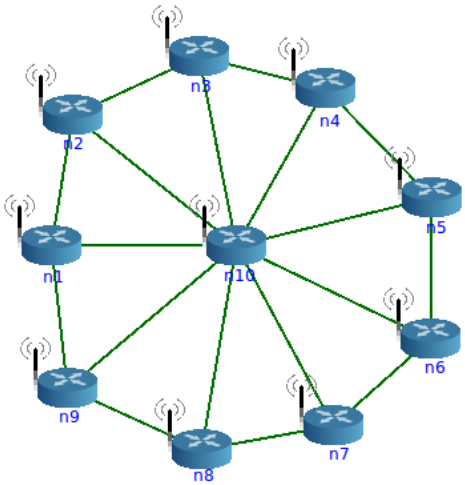

Figure 10. Wheel topology.

Eventually, this work will investigate whether survivability predictions extend to mobile scenarios. At first glance, it seems likely; when nodes move, the topologies change. It may be the case that survivability predictions can be made for each topology formed during the movement.

Attack. The types of attacks are spoofing and forwarding, as described in the previous section. For the automated process, the attack scripts take additional inputs, start time and duration.

Attack node number. This indicates which of the 10 nodes in the scenario will issue an attack.

Parameters that were controlled across all emulation instances are discussed below.

Attack time. This parameter indicates how long a node must wait before executing the attack. This is set to 60 seconds.

Scenario duration. Each scenario is broken into three phases—before, during, and after an attack was issued. Each scenario is 3 min long, divided into 60 s phases.

Data flow. During each scenario, nodes communicate using TCP and UDP data packets. Packets are 1280 bytes in size and are sent 50 times per second. The traffic is generated using mgen (*26*). Each node opens six sockets, three outgoing and three incoming. Table 1 contains the data flows that are used during each instance.

Table 1. Traffic Data flows between nodes. Column 1 indicates the source node while columns 2 and 3 indicate the nodes being sent TCP and UDP packets respectively.

Node	TCP Outgoing	UDP Outgoing
1	**10**	**2,3**
2	**1**	**3,4**
3	**2**	**4,5**
4	**3**	**5,6**
5	**4**	**6,7**
6	**5**	**7,8**
7	**6**	**8,9**
8	**7**	**9,10**
9	**8**	**10,1**
10	**9**	**1,2**

Attack Duration. The duration of attack is a constant 60 seconds.

4.2 Log Data Collection

During each scenario all nodes log incoming data. Depending on the type of node (legitimate or attacker), different attributes are logged.

Attacker Node Logs

Attacker nodes run the tshark process using the following flags shown in figure 11:

```
tshark   -i <ifx> -T fields -E separator=, -e frame.time_epoch -e frame.len -e
         frame.protocols  -e ip.src -e ip.dst –e ipv6.src -e ipv6.dst -e tcp.srcport -e tcp.dstport
         -e udp.srcport   -e udp.dstport -l
```

Figure 11. Attacker log tshark flags.

The output from tshark is piped into a python script. Each second, the collected data are averaged and written to memory. The attributes collected by the attacker are listed below.

Time. The time stamp indicating when the data are captured.

Flows. This attribute contains information about promiscuous traffic that is seen passing through the attacker node. This does not include packets where the attacker is either the source or destination IP address. The flows contain either TCP or UDP as the traffic type. The hop count from the attacker to the source address and the hop count from the attacker to the destination address are also captured.

Routes. A dump of the routing tables using the Linux *route* command.

Attack running. The name of the attack that is currently running; if no attack is running, this attribute is empty.

Legitimate Node Logs

Traffic flows are generated using mgen. Features of mgen allow trivial collection of statistics such as delays (using timestamps within messages), and number of received, missed and out of order packets (using sequence numbers). mgen is executed with the following flags shown in figure 12.

```
mgen flush input <pathToFlow> output /dev/null
```

Figure 12. Non-Attacker log mgen flags.

Legitimate nodes log routes, timestamps, and the attack running in the same way as the attacker. The attack running attribute in the legitimate node logs is used solely for synchronization purposes. Flows are collected in a similar fashion as the attacker, except that IP addresses are used instead of hops. All of these attributes are logged to a file and used later to form a network state from the attacker's viewpoint, as described in the next section.

5. Network Representation

As the number of nodes in a network increases, so does the complexity of analyzing the impact of an attack on the network. Part of the reason for this lies in the fact that representing a network and the traffic flows is difficult and easily fall victim to state-space explosion.

To avoid this, instead of representing the network as a collection of source and destination IP addresses, the distance (hops) from the attacker's location are used. Figure 13 shows an example of this. In the sample, node n1 is sending packets to n3 and n3 is sending packets to n4 (denoted by dotted lines). From the attacker's view, the n1 to n3 communication is seen as hop (1) to hop (1) with passthrough. The n3 to n4 communication is seen as hop (1) to hop (2) with no passthrough.

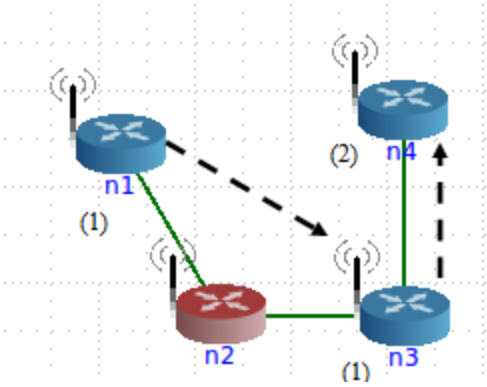

Figure 13. Representation by hops. Hop counts are labeled in parentheses and dotted lines indicate traffic.

The network representation is defined as the collection of flow descriptions over an entire emulation instance. Flow descriptions are composed of the parameters in table 2. These flow descriptions were captured by fusing the legitimate and attacker log files. Log files are synchronized using the attack name.

Table 2. Network representation parameters.

#	Attribute	Description	Capture Source
1	fromHop	Hops from the attacker node to the source.	LegFlw+AttRte
2	toHop	Hops from the attacker node to the destination.	LegFlw+AttRte
3	dataType	Data, not control, packet type.	Leg Flw
4	distanceTraveled	Hops from source to destination.	Leg Flw
5	passThrough	Whether this flow pass through the attacker.	LegFlw,Rte+AttRte
6	beforeStats	Mgen data before an attack.	Leg Flw
7	duringStats	Mgen data during an attack.	Leg Flw
8	afterStats	Mgen data after an attack.	Leg Flw
9	attackName	Spoofing or forwarding indicator.	Leg+Att
10	duringLinkLost	Whether a link is lost during an attack.	LegFlw
11	srcIsSpoofed	Whether the source address is spoofed.	LegFlw+AttRte
12	destIsSpoofed	Whether the destination address is spoofed.	LegFlw+AttRte
13	hopsSpoofedToDest	Hops from the spoofed to the destination.	LegFlw+AttRte
14	spoofedBetweenAttacker	Whether the spoofed is between the attacker and the destination.	LegFlw,Rte+AttRte
15	spoofedBetween AttackerGW	Whether the spoofed is a gateway (directly connected) node on the path to the destination.	LegFlw,Rte+AttRte
16	destBetween SpoofedAndAttacker	Whether the destination is between the spoofed and the attacker.	LegFlw,Rte+AttRte
17	destBetween SpoofedAndAttackerGW	Whether the destination is a gateway node on the path to the attacker.	LegFlw,Rte+AttRte
18	attackerBetween SpoofedAndDest	Whether the destination is between the spoofed and the attacker.	LegFlw,Rte+AttRte
19	attackerBetween SpoofedAndDestGW	Whether the destination is a gateway node on the path to the attacker.	LegFlw,Rte+AttRte
20	srcBetween SpoofedAndDest	Whether the destination is between the spoofed and the attacker.	LegFlw,Rte+AttRte
21	srcBetween SpoofedAndDestGW	Whether the destination is a gateway node on the path to the attacker.	LegFlw,Rte+AttRte
22	altPathWithoutAttacker	Whether an alternate path between source and destination exists without the attacker.	LegFlw,Rte+AttRte

In table 2, parameters with a capture source value *Leg Flw* are taken from the flows in the legitimate log file. More specifically, parameters 6−8 encapsulate information about missed, out of order, and total packets received along with delays. The *Leg Flw+Att Rte* value indicates that the parameters are captured by using the IP addresses in the legitimate log file flows. Next these are cross-referenced with the routes in the attacker log file. The *LegFlw,Rte+AttRte* capture sources are obtained by first identifying a flow from the legitimate log file. Paths are traced by iteratively following the gateway nodes, provided by the routes in the log files. As a special case, parameter 22 is derived by using the python networkx package (*27*). Using networkx, a graph data structure is built from node connections as given by route tables. The existence of an alternate path is determined after deletion of the attacker node in the data structure. In the case of forwarding attacks, parameters 11–22 are always false.

The selection of parameters was an iterative process. Initially only parameters 1–10 were used; the others were chosen after a deeper analysis. This deeper analysis consisted of the several steps. First, the data were captured and represented using the initial parameters. A python script, *conflictDetect.py,* generated a hash table or dictionary using all parameters, except *duringLinkLost*, as keys in the key/value pair. The Boolean parameter (taking on either true or false) *duringLinkLost* was the value in the key/value pair. The python script went through each network representation. If a collision was found and *duringLinkLost* differed, these were considered conflicting flows.

For each conflicting flow, the number of times that the *duringLinkLost* parameter resulted as *true* and *false* was stored. In the case where there was an equal amount of *true* and *false* counts, the emulation instances associated with the flows were run again. In the case where the counts were not equal, the emulation instances associated with the minority were run again. Sometimes the emulation instance encountered an unknown error and all links randomly disconnected.

More often, the reason for the conflicts resulted due to a lack of representation of some network characteristic. Analysis of these cases led to the additional parameters 11−20. The network representation was used to train a classifier to predict network survivability, specifically the *duringLinkLost* parameter.

6. Evaluation

An experiment was conducted to validate the hypothesis that survivability of links in a network under spoofing and forwarding attacks can be predicted. The experiment used the dataset described in the Data Collection Section (section 4). The dataset was formatted into the network representation described in the Network Representation (section 5).

To determine the quality of the network representation, the numbers of conflicts (as described in the Network Representation (section 5) were counted. Next, a predictor was built using REPTree as implemented in the WEKA (*28*) data-mining toolset. The test method used was 10 fold cross-validation, with this method 90% of the data are set as training to predict the remaining 10%. This process is repeated 10 times using different portions of data.

A subset of the parameters was used as training attributes to predict *duringLinkLost.* Initially, parameters 1−5 and 9 were used (called the partial set) and then parameters 11−22 were added (called the all set) to determine improvements. Attributes 7 and 8 were not predicted because the purpose of this experiment was to determine if link loss can be predicted, not to determine numeric degradation. This would require more varied flow characteristics (in the current dataset all flows are equal in rate and size).

In general, the dataset used for evaluation consists of all combinations of the following configurations:

- Routing Protocols: OSPFv3MDR, OLSR
- Topologies: chain, connected_grid, cycle, star, tree, two-centroid, and wheel
- Attacks: forwarding, spoofing

Additionally, there are 10 nodes with 3 outgoing connections (2 UDP and 1 TCP). Each emulation instance contains one attacking node selected using a round-robin approach. In total there are 32626 flows, 17251 with OLSR and 15375 with OSPFv3MDR. There were a total of 3115 unique flows with OLSR and 2526 with OSPFv3MDR. In very few cases, a malfunction in CORE caused some nodes to stop capturing data; as a result, the dataset contains a small amount of noise.

7. Results

Table 3 shows only a small percentage of the flows in the dataset conflicted. This is strong evidence that a classifier can be derived that will predict, given the network representation, whether certain connections will be lost when a forwarding attack or a spoofing attack occur. Reasons for the conflicts at the time of this writing may be due to malfunctions in the emulator or it may be the case that further venturing into the specifics of the routing protocol is required (route ordering, source code investigation, etc.).

Table 3. Percentage of flows that conflicted per protocol.

Protocol	Attack	% Conflict
OLSR	Forwarding	0.06
	Spoofing	2.00
OSPFv3MDR	Forwarding	0.00
	Spoofing	0.03

Four REPTree classifiers were trained (OLSR forwarding, OLSR spoofing, OSPF forwarding, OSPF spoofing) using 10 fold cross-validation. Tables 4 and 5 contain the results for OLSR and OSPF configuration respectively.

Table 4. Weighted averages for classification of *duringLinkLost* with OLSR.

Attack	Parameters Used	True Positive	False Positive	F-Measure
Forwarding	Partial	0.998	0.018	0.998
	All	0.998	0.018	0.998
Spoofing	Partial	0.975	0.161	0.975
	All	0.983	0.103	0.983

Table 5. Weighted averages for classification of *duringLinkLost* with OSPFv3MDR.

Attack	Parameters Used	True Positive	False Positive	F-Measure
Forwarding	Partial	1	0	1
	All	1	0	1
Spoofing	Partial	0.997	0.248	0.991
	All	0.998	0.031	0.998

The results using cross-validations are a good indication that the classifier will do well with unseen scenarios. In the case of OLSR, although there were a higher number of conflicts, the REPTree still performed reasonably well. Using the full set of parameters generally improved the prediction, reducing the false positives, of spoofing impacts.

An interesting note is that when attempting to augment the training set with parameter 6 (before-attack statistics) the classifier did not improve.

The generated predictor models using all parameters are provided in figures 14–20. The predicting link survivability during forwarding attacks is trivial; however, this is not the case with spoofing attacks.

```
passthrough = true : true
passthrough = false
|   distance = 1 : false
|   distance = 2 : false
|   distance = 3
|   |   type = TCP
|   |   |   toHop = 1
|   |   |   |   fromHop = 1 : false
|   |   |   |   fromHop = 2 : true
|   |   |   |   fromHop = 3 : false
|   |   |   |   fromHop = 4 : false
|   |   |   |   fromHop = 5 : false
|   |   |   |   fromHop = 6 : false
|   |   |   |   fromHop = 7 : false
|   |   |   |   fromHop = 8 : false
|   |   |   |   fromHop = 9 : false
|   |   |   |   fromHop = 10 : false
|   |   |   toHop = 2 : false
|   |   |   toHop = 3 : false
|   |   |   toHop = 4 : false
|   |   |   toHop = 5 : false
|   |   |   toHop = 6 : false
|   |   |   toHop = 7 : false
|   |   |   toHop = 8 : false
|   |   |   toHop = 9 : false
|   |   |   toHop = 10 : false
|   |   type = UDP : false
|   distance = 4 : false
|   distance = 5 : false
|   distance = 6 : false
|   distance = 7 : false
|   distance = 8 : false
|   distance = 9 : false
|   distance = 10 : false
```

Figure 14. OLSR Forwarding survivability model.

```
destSpoofed = true
|   isDstBetweenSpoofedAndAttacker = true
|   |   toHop = 1
|   |   |   fromHop = 1 : false
|   |   |   fromHop = 2 : false
|   |   |   fromHop = 3 : true
|   |   |   fromHop = 4 : true
|   |   |   fromHop = 5 : true
|   |   |   fromHop = 6 : false
|   |   |   fromHop = 7 : false
|   |   |   fromHop = 8 : false
|   |   |   fromHop = 9 : true
|   |   |   fromHop = 10 : false
|   |   toHop = 2 : false
|   |   toHop = 3 : false
|   |   toHop = 4 : false
|   |   toHop = 5 : false
|   |   toHop = 6 : false
|   |   toHop = 7 : false
|   |   toHop = 8 : false
|   |   toHop = 9 : false
|   |   toHop = 10 : false
|   isDstBetweenSpoofedAndAttacker = false
|   |   fromHop = 1 : true
|   |   fromHop = 2
|   |   |   toHop = 1 : true
|   |   |   toHop = 2
|   |   |   |   distance = 1 : false
|   |   |   |   distance = 2
|   |   |   |   |   type = TCP : true
|   |   |   |   |   type = UDP
|   |   |   |   |   |   isDstBetweenSpoofedAndAttackergw = true : false
|   |   |   |   |   |   isDstBetweenSpoofedAndAttackergw = false : true
|   |   |   |   distance = 3 : true
|   |   |   |   distance = 4 : false
|   |   |   |   distance = 5 : false
|   |   |   |   distance = 6 : false
|   |   |   |   distance = 7 : false
|   |   |   |   distance = 8 : false
|   |   |   |   distance = 9 : false
|   |   |   |   distance = 10 : false
|   |   |   toHop = 3 : true
|   |   |   toHop = 4 : true
|   |   |   toHop = 5 : true
|   |   |   toHop = 6 : true
|   |   |   toHop = 7 : true
|   |   |   toHop = 8 : true
|   |   |   toHop = 9 : true
|   |   |   toHop = 10 : true
|   |   fromHop = 3 : true
|   |   fromHop = 4 : true
|   |   fromHop = 5 : true
|   |   fromHop = 6 : true
|   |   fromHop = 7 : true
|   |   fromHop = 8 : true
|   |   fromHop = 9 : true
|   |   fromHop = 10 : true
```

Figure 15. OLSR Spoofing survivability model part 1.

```
destSpoofed = false
|   srcSpoofed = true
|   |   type = TCP
|   |   |   spoofedBetweenAttackergw = true : false
|   |   |   spoofedBetweenAttackergw = false
|   |   |   |   toHop = 1 : true
|   |   |   |   toHop = 2
|   |   |   |   |   fromHop = 1 : true
|   |   |   |   |   fromHop = 2
|   |   |   |   |   |   distance = 1 : false
|   |   |   |   |   |   distance = 2 : true
|   |   |   |   |   |   distance = 3 : true
|   |   |   |   |   |   distance = 4 : true
|   |   |   |   |   |   distance = 5 : false
|   |   |   |   |   |   distance = 6 : false
|   |   |   |   |   |   distance = 7 : false
|   |   |   |   |   |   distance = 8 : false
|   |   |   |   |   |   distance = 9 : false
|   |   |   |   |   |   distance = 10 : false
|   |   |   |   |   fromHop = 3 : true
|   |   |   |   |   fromHop = 4 : true
|   |   |   |   |   fromHop = 5 : true
|   |   |   |   |   fromHop = 6 : true
|   |   |   |   |   fromHop = 7 : true
|   |   |   |   |   fromHop = 8 : true
|   |   |   |   |   fromHop = 9 : true
|   |   |   |   |   fromHop = 10 : true
|   |   |   |   toHop = 3 : true
|   |   |   |   toHop = 4 : true
|   |   |   |   toHop = 5 : true
|   |   |   |   toHop = 6 : true
|   |   |   |   toHop = 7 : true
|   |   |   |   toHop = 8 : true
|   |   |   |   toHop = 9 : true
|   |   |   |   toHop = 10 : true
|   |   type = UDP
|   |   |   passthrough = true : true
|   |   |   passthrough = false : false
|   srcSpoofed = false
|   |   distance = 1 : false
|   |   distance = 2 : false
|   |   distance = 3 : false
|   |   distance = 4
|   |   |   hopsToSpoofed = 0 : false
|   |   |   hopsToSpoofed = 1
|   |   |   |   passthrough = true
|   |   |   |   |   isDstBetweenSpoofedAndAttacker = true : true
|   |   |   |   |   isDstBetweenSpoofedAndAttacker = false
|   |   |   |   |   |   spoofedBetweenAttackergw = true : true
|   |   |   |   |   |   spoofedBetweenAttackergw = false : false
|   |   |   |   passthrough = false : false
|   |   |   hopsToSpoofed = 2 : false
|   |   |   hopsToSpoofed = 3 : false
|   |   |   hopsToSpoofed = 4 : false
|   |   |   hopsToSpoofed = 5 : false
|   |   |   hopsToSpoofed = 6 : false
|   |   |   hopsToSpoofed = 7 : false
|   |   |   hopsToSpoofed = 8 : false
|   |   |   hopsToSpoofed = 9 : false
|   |   |   hopsToSpoofed = 10 : false
```

Figure 16. OLSR Spoofing survivability model part 2.

```
|   |   distance = 5
|   |   |   hopsToSpoofed = 0 : false
|   |   |   hopsToSpoofed = 1
|   |   |   |   passthrough = true
|   |   |   |   |   isDstBetweenSpoofedAndAttacker = true : true
|   |   |   |   |   isDstBetweenSpoofedAndAttacker = false
|   |   |   |   |   |   spoofedBetweenAttacker = true : false
|   |   |   |   |   |   spoofedBetweenAttacker = false : true
|   |   |   |   passthrough = false : false
|   |   |   hopsToSpoofed = 2 : false
|   |   |   hopsToSpoofed = 3 : false
|   |   |   hopsToSpoofed = 4 : false
|   |   |   hopsToSpoofed = 5 : false
|   |   |   hopsToSpoofed = 6 : false
|   |   |   hopsToSpoofed = 7 : false
|   |   |   hopsToSpoofed = 8 : false
|   |   |   hopsToSpoofed = 9 : false
|   |   |   hopsToSpoofed = 10 : false
|   |   distance = 6 : false
|   |   distance = 7 : false
|   |   distance = 8
|   |   |   hopsToSpoofed = 0 : false
|   |   |   hopsToSpoofed = 1 : true
|   |   |   hopsToSpoofed = 2 : false
|   |   |   hopsToSpoofed = 3 : false
|   |   |   hopsToSpoofed = 4 : false
|   |   |   hopsToSpoofed = 5 : false
|   |   |   hopsToSpoofed = 6 : false
|   |   |   hopsToSpoofed = 7 : false
|   |   |   hopsToSpoofed = 8 : false
|   |   |   hopsToSpoofed = 9 : false
|   |   |   hopsToSpoofed = 10 : false
|   |   distance = 9
|   |   |   hopsToSpoofed = 0 : false
|   |   |   hopsToSpoofed = 1 : true
|   |   |   hopsToSpoofed = 2 : false
|   |   |   hopsToSpoofed = 3 : false
|   |   |   hopsToSpoofed = 4 : false
|   |   |   hopsToSpoofed = 5 : false
|   |   |   hopsToSpoofed = 6 : false
|   |   |   hopsToSpoofed = 7 : false
|   |   |   hopsToSpoofed = 8 : false
|   |   |   hopsToSpoofed = 9 : false
|   |   |   hopsToSpoofed = 10 : false
|   |   distance = 10 : false
```

Figure 17. OLSR Spoofing survivability model part 3.

```
passthrough = true : true
passthrough = false : false
```

Figure 18. OSPFv3MDR Forwarding survivability model.

```
passthrough = true
|   isDstBetweenSpoofedAndAttacker = true
|   |   hopsToSpoofed = 0 : true
|   |   hopsToSpoofed = 1 : true
|   |   hopsToSpoofed = 2 : true
|   |   hopsToSpoofed = 3 : true
|   |   hopsToSpoofed = 4
|   |   |   fromHop = 1 : false
|   |   |   fromHop = 2 : false
|   |   |   fromHop = 3 : false
|   |   |   fromHop = 4 : true
|   |   |   fromHop = 5 : false
|   |   |   fromHop = 6 : false
|   |   |   fromHop = 7 : false
|   |   |   fromHop = 8 : false
|   |   |   fromHop = 9 : false
|   |   |   fromHop = 10 : false
|   |   hopsToSpoofed = 5
|   |   |   fromHop = 1 : false
|   |   |   fromHop = 2 : false
|   |   |   fromHop = 3 : false
|   |   |   fromHop = 4 : false
|   |   |   fromHop = 5 : true
|   |   |   fromHop = 6 : false
|   |   |   fromHop = 7 : false
|   |   |   fromHop = 8 : false
|   |   |   fromHop = 9 : false
|   |   |   fromHop = 10 : false
|   |   hopsToSpoofed = 6
|   |   |   fromHop = 1 : false
|   |   |   fromHop = 2 : false
|   |   |   fromHop = 3 : false
|   |   |   fromHop = 4 : false
|   |   |   fromHop = 5 : false
|   |   |   fromHop = 6 : true
|   |   |   fromHop = 7 : false
|   |   |   fromHop = 8 : false
|   |   |   fromHop = 9 : false
|   |   |   fromHop = 10 : false
|   |   hopsToSpoofed = 7 : false
|   |   hopsToSpoofed = 8 : true
|   |   hopsToSpoofed = 9 : true
|   |   hopsToSpoofed = 10 : true
|   isDstBetweenSpoofedAndAttacker = false
```

Figure 19. OSPFv3MDR Spoofing survivability model part 1.

```
OSPF spoofing p2
|   |   destSpoofed = true : true
|   |   destSpoofed = false
|   |   |   isDstBetweenSpoofedAndAttackergw = true
|   |   |   |   hopsToSpoofed = 0 : false
|   |   |   |   hopsToSpoofed = 1 : false
|   |   |   |   hopsToSpoofed = 2
|   |   |   |   |   isAttackerBetweenSpoofedAndDstgw = true : true
|   |   |   |   |   isAttackerBetweenSpoofedAndDstgw = false : false
|   |   |   |   hopsToSpoofed = 3
|   |   |   |   |   isAttackerBetweenSpoofedAndDstgw = true : true
|   |   |   |   |   isAttackerBetweenSpoofedAndDstgw = false : false
|   |   |   |   hopsToSpoofed = 4 : false
|   |   |   |   hopsToSpoofed = 5 : false
|   |   |   |   hopsToSpoofed = 6 : false
|   |   |   |   hopsToSpoofed = 7 : false
|   |   |   |   hopsToSpoofed = 8 : false
|   |   |   |   hopsToSpoofed = 9 : false
|   |   |   |   hopsToSpoofed = 10 : false
|   |   |   isDstBetweenSpoofedAndAttackergw = false : false
passthrough = false
|   destSpoofed = true
|   |   isDstBetweenSpoofedAndAttackergw = true : false
|   |   isDstBetweenSpoofedAndAttackergw = false
|   |   |   fromHop = 1 : false
|   |   |   fromHop = 2 : true
|   |   |   fromHop = 3
|   |   |   |   distance = 1 : false
|   |   |   |   distance = 2 : false
|   |   |   |   distance = 3 : false
|   |   |   |   distance = 4 : true
|   |   |   |   distance = 5 : true
|   |   |   |   distance = 6 : true
|   |   |   |   distance = 7 : false
|   |   |   |   distance = 8 : false
|   |   |   |   distance = 9 : false
|   |   |   |   distance = 10 : false
|   |   |   fromHop = 4 : false
|   |   |   fromHop = 5 : false
|   |   |   fromHop = 6 : false
|   |   |   fromHop = 7 : false
|   |   |   fromHop = 8 : false
|   |   |   fromHop = 9 : false
|   |   |   fromHop = 10 : false
|   destSpoofed = false : false
```

Figure 20. OSPFv3MDR Spoofing survivability model part 2.

8. Conclusions and Future Work

The work provided in this report has benefits in fundamental processes of survivability analysis. Specifically, this report shows that it is possible to predict the survivability of network flows, unlike work in the past that focuses only on throughput totals. Experiment results show that by representing a network from an attacker's perspective, link loss due to spoofing and data forwarding attacks can be accurately predicted.

The following are areas for future work. One important endeavor is determining whether the methods used here will apply to mobile nodes. This seems likely since during a mobile scenario, nodes form topologies that can be formatted into the network representation described in the Network Representation (section 5). Next, testing will be conducted with other ad-hoc protocols such as ad hoc on-demand distance vector (AODV), routing information protocol (RIP), and better approach to mobile ad hoc network (BATMAN) along with defense mechanisms. Whether it is possible to predict numeric impacts such as delay, missed packets, time until impact, recovery after attack, and others will be determined.

This work used TCP and UDP packets, but in the future the effects of different traffic will be investigated. Measuring the survivability given a wider range of attacks, such as route fabrication and multi-attacker scenarios, is planned. Future work will determine whether the impact of these attacks can be accurately predicted. To improve the accuracy of the methods, layer-1 and layer-2 emulation will be tested using such tools as EMANE and the available plugins for communication loss and radio models.

As many systems currently use OPNet for network simulation, a future engineering effort will convert OPNet scenarios into a format that can be read by CORE, hence allowing the use of the survivability analysis described in this report.

While this work focuses on malicious nodes, it does not take into account probabilities of nodes becoming targets. Future work will augment attack graph and attack grammar data to provide this feature.

9. References

1. Yi, J. http://www.jiaziyi.com/documents/20080229_A_Survey_on_the_Applications_of_MANET.pdf, A Survey on the Applications of MANET. *Architecture*.
2. Wu, B.; Chen, J.; Wu, J.; Cardei, M. A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks. *Signals and Communication Technology*, Special Issue: *Wireless Network Security* **2007**, 103–135.
3. Ambhaikar, A.; Mitra, D.; Deshmukh, R. Performance of MANET Routing Protocol for Improving Scalability. *International Journal of Advanced Engineering Application* **2011**, January, 15–18.
4. Britton, M.; Coyle, *A. Performance Analysis of the BATMAN Wireless Ad-Hoc Network Routing Protocol with Mobility and Directional Antennas*; SANLAB Technical Report; 2011.
5. Tuteja, A.; Gujral, R.; Thalia. S. Comparative Performance Analysis of DSDV, AODV and DSR Routing Protocols in MANET using NS2. *International Conference on Advances in Computer Engineering (ACE)*, Taipei, Taiwan, November 2010.
6. Andel, T. R.; Yasinsac, A. Surveying Security Analysis Techniques in MANET Routing Protocols. *IEEE Communications Surveys Tutorials* **2007**, *9*, 70–84.
7. Hu, Y. C.; Perrig, A.; Johnson, D. B. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks* **2005**, *11* (1–2), 21–38.
8. Hu, Y. C.; Johnson, D. B.; Perrig, A. 8: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks* **2003**, *1* (1), 175–192.
9. Perrig, A.; Canetti, R.; Tygar, J. D.; Song, D. The 9 Broadcast Authentication Protocol. *CryptoBytes*, *5*, (2 – Summer/Fall).
10. Lu, S.; Li, L.; Lam, K. Y.; Jia, L. 10: A MANET routing protocol that can withstand black hole attack. *International Conference on Computational Intelligence and Security*, Bejing, China, December 11–14, 2009, 421–425.
11. Moustafa, M. A.; Youssef, M. A.; El-Derini, M. N. 11: A Multipath Secure Reliable Routing Protocol For WSNs. *International Conference on Computer Systems and Applications (AICCSA)*, Sharm El-Sheikh, Egypt, December 27–30, 2011.
12. Stavrou, E.; Pitsillides, A. A Survey On Secure Multipath Routing Protocols in WSNs. *Computer Networks* **2010**, *54* (13), 2215–2238.

13. Parsons, M.; Ebinger, P. Performance Evaluation of the Impact of Attacks on mobile Ad-Hoc networks. *International Symposium on Reliable Distributed Systems*, Niagra Falls, NY, September 27–30, 2009.

14 Kiddie, P. D. Decentralised Soft-Security in Distributed Systems. *Ph.D. Thesis, University of Birmingham*, 2011.

15. Ochola, E. O.; Eloff, M. M. A Review of Black Hole Attack on AODV Routing in MANET. *Information Security, South Africa*, 2011.

16. Andel, T. R.; Yasinsac, A. On the Credibility of MANET Simulations. *IEEE Computer* **2006**, *39* (7).

17. Lima, M.; Dos Santos, A.; Pujolle, G. A Survey of Survivability in Mobile Ad Hoc Networks. *Communications Surveys & Tutorials* **2009**, *11* (1), 66–77.

18. Ellison, R.; Fisher, D.; Linger, R.; Lipson, H.; Longstaff, T.; Mead, N. *Survivable Network Systems: An Emerging Discipline* ; cmu/sei-97-tr-013; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, 1997.

19. Kim, K.; Roh, B.; Ko, Y. B.; Choi, W. J.; Son, E. S. Survivability Measure For Multichannel MANET-Based Tactical Networks, *Advanced Communication Technology*, Seoul, Korea, 13–16 February 2011.

20. Wang, T.; Huang, C. H.; Xiang, K.; Zhou, K. X. Survivability Evaluation for MANET Based on Path Reliability. *Networks Security Wireless Communications and Trusted Computing***.** Wuhan, Hubei, China, April 24–25, 2010, *1*, 378–381.

21. Wang, J.; Yu, Z. Research on Quantitative Analysis Model of MANET Survivability. *Electrical and Control Engineering*, Yichang, China, September 16–18, 2011.

22. Wang, T.; Huang, C. H. A Neural Network Model for Evaluating Mobile Ad Hoc Wireless Network Survivability. *Advances in Neural Networks*, Shanghai, China, June 2010.

23. Ahrenholz, J.; Danilov, C.; Henderson, T. R.; Kim, J. H. CORE: A Real-Time Network Emulator. *Military Communications Conference*, San Diego, CA, November 17–19, 2008.

24. Jain, K.; Roy-Choudhary, A.; Somasundaram, K. K.; Wang, B.; Baras, J. S. Studying Real-time Traffic in Multi-hop Networks Using the EMANE Emulator: Capabilities and Limitations. *Digital Repository at the University of Maryland (TR_2011-01)*, 2011.

25. http://labs.cengen.com/emane/download/addons/commeffect.html (accessed February 14, 2012).

26. http://cs.itd.nrl.navy.mil/work/mgen/ (accessed February 14, 2012).

27. Hagberg, A.; Schult, D.; Swart, P. NetworkX Library developed at the Los Alamos National Laboratory Labs Library (DOE) by the University of California. Code available at https://networkx.lanl.gov (accessed February 14, 2012).

28. Hall, Mark; Frank, Eibe; Holmes, Geoffrey; Pfahringer, Bernhard; Reutemann, Peter; Witten, Ian H. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* **2009**, *11* (1), 11.

List of Symbols, Abbreviations, and Acronyms

AODV	ad hoc on-demand distance vector
BATMAN	better approach to mobile ad hoc network
CORE	common open research emulator
EMANE	extendable mobile ad hoc network emulator
HNA	host network announcement
IP	Internet Protocol
NRLOLSR	Naval Research Laboratory optimized link state routing
OLSR	optimized link state routing
RIP	routing information protocol
TCP	transmission control protocol
UDP	user datagram protocol

1	ADMNSTR
ELEC	DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1 CD 1 WORD VERSION	US ARMY RSRCH LAB MELE ASSOCIATES ATTN RDRL SLE E D NEVAREZ BLDG 1622 RM 216 WHITE SANDS MISSILE RANGE NM 88002-5501
1 CD 1 HC	US ARMY RSRCH LAB ATTN RDRL SLE I J ACOSTA BLDG 1646 WHITE SANDS MISSILE RANGE NM 88002-5513
3 HCs	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIO LL TECHL LIB ATTN RDRL CIO LT TECHL PUB ADELPHI MD 20783-1197
1 HC	DANIEL LANDIN ARMY RESEARCH LABORATORY ATTN RDRL SLE I BLDG 1624 WSMR NM 88002-5513

Total: 9 (1 PDF, 2 CDs, 1 WORD VERSION, 5 HCs)

INTENTIONALLY LEFT BLANK.