ABSTRACT

MEREDITH, JAY RUSSELL. Increasing Network Resilience to Persistent OSPFv2 Attacks through Automated Network Recovery. (Under the direction of Dr. Rudra Dutta and Dr. Douglas Reeves.)

Networked systems are under constant attack and, inevitably, will be compromised. Therefore, it is critical that routing protocols be more resilient, in order to recover from malicious actors. The cost of not preparing for recovery, both financial and in terms of data loss, far outweigh any cost incurred preparing for recovery.

In this dissertation we explore the importance of recovery as a key component of making networks more resilient against malicious actors. We first discuss the need for added resilience to OSPFv2 networks by reviewing the five primary accepted recovery techniques (Restore, Rollforward, Reinitialize, Reconfigure, and Replace) and then experimentally evaluate the success of each technique against modern persistent OSPFv2 attacks. Next, we further synthesize recovery and resilience to form the Virtual Router Resiliency Cluster (VRRC), a system designed to seamlessly integrate with any existing OSPFv2 network, making the network resilient to all known link-state attacks against OSPFv2 in addition to providing a rudimentary network recovery mechanism.

Having laid the foundation for network recovery, we present two new approaches for triggering auto-recovery: through fingerprinting routers and using time as a means of authenticating LSA updates. Routers can “fingerprint” each other after the network is initially brought up in a trusted environment. If the fingerprint of a router changes, it may indicate that a malicious actor is spoofing or that the router is compromised. Detecting this change can then be used to trigger a recovery mechanism. Our second approach demonstrates how the LSA Synchronized Time Controller (LSTC) makes the network more resilient to attacks against OSPFv2. Routers can agree, in advance and out of band, on a very specific sequence of intervals at which they will generate LSAs, and treat any LSA that is generated at any other time as attacks - therefore simply ignoring them and if necessary triggering a recovery mechanism. The lessons learned from these techniques motivate recovery as a key component in creating more resilient network against malicious actors.
Increasing Network Resilience to Persistent OSPFv2 Attacks through Automated Network Recovery

by
Jay Russell Meredith

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2019

APPROVED BY:

Dr. Laurie Williams
Dr. Alex Kapravelos

Dr. Rudra Dutta
Co-chair of Advisory Committee
Dr. Douglas Reeves
Co-chair of Advisory Committee
DEDICATION

To my family.
# Table of Contents

## List of Tables

- vii

## List of Figures

- viii

## Chapter 1 Introduction

- 1.1 Resilience
- 1.2 Thesis Statement
- 1.3 Contributions
- 1.4 Dissertation Outline

## Chapter 2 Background

- 2.1 TCP/IP and the Routing Infrastructure
- 2.2 OSPFv2
  - 2.2.1 OSPFv2 Operations
  - 2.2.2 Strengths/Weaknesses
- 2.3 Routing Attacks
- 2.4 Specific Attacks Against OSPFv2
  - 2.4.1 OSPFv2 Vulnerabilities
  - 2.4.2 Table Overflow Attack
  - 2.4.3 LSID
  - 2.4.4 Max-Age
  - 2.4.5 Max-Seq Number
  - 2.4.6 Seq++
  - 2.4.7 Remote False Adjacency
  - 2.4.8 Disguised LSA v.1
  - 2.4.9 Disguised LSA v.2
  - 2.4.10 Partition Attack
  - 2.4.11 Loki
- 2.5 Brief Summary of Proposed Countermeasures

## Chapter 3 Related Work

- 3.1 Revised Network Resilience Architecture
- 3.2 Review of Resilience Architectures
  - 3.2.1 Cyber Resilience Engineering Framework (CREF)
  - 3.2.2 Resili-Net
  - 3.2.3 \( D^2R^2 + DR \)
  - 3.2.4 Building Secure, Resilient Architectures for Cyber Mission Assurance
- 3.3 Revised Resilience Architecture
  - 3.3.1 Anticipate
  - 3.3.2 Prevent
  - 3.3.3 Detect
  - 3.3.4 Recover
  - 3.3.5 Adapt
3.4 Recovery Terminology and Characteristics ........................................... 27
  3.4.1 Degree of Automation ................................................................. 27
  3.4.2 Response Time ............................................................................. 28
  3.4.3 Distributed .................................................................................... 28
  3.4.4 Level of Operation/Deployment ...................................................... 28
  3.4.5 Attack-Adaptive ........................................................................... 29
  3.4.6 IDS Reliance ................................................................................ 29
  3.4.7 Response Selection ................................................................. 29
  3.4.8 Recovery Technique ........................................................... 30
3.5 Review of Network Recovery Techniques ............................................. 30
  3.5.1 Manual Recovery ................................................................. 33
  3.5.2 Damage Control and Assessment (DC&A) ...................................... 33
  3.5.3 Cooperative Security Manager (CSM) ............................................ 34
  3.5.4 Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) ........................................................................... 34
  3.5.5 Lee’s IRS .................................................................................... 34
  3.5.6 Adaptive Agent-based Intrusion Response System (AAIRS) .............. 35
  3.5.7 Cooperative Intrusion Traceback and Response Architecture (CITRA) ........................................................................... 35
  3.5.8 Network IRS ............................................................................... 35
  3.5.9 Specification-based IRS ............................................................ 36
  3.5.10 Phoenix .................................................................................... 36
  3.5.11 Distributed Backup for Local Area Networks (DIBS) ...................... 36
  3.5.12 Adaptive Intrusion Tolerant System (ADEPTS) ................................ 37
  3.5.13 Distributed Intrusion Prevention System (DIPS) .......................... 37
  3.5.14 Jahnke ................................................................................... 37
  3.5.15 Fast Proactive Recovery from Concurrent Failures (FPRCF) ............. 38
  3.5.16 The Fault/intrusion REmoVal through Evolution and Recovery (FOREVER) ........................................................................... 38
  3.5.17 Response Recovery Engine (RRE) ................................................ 38
  3.5.18 Spontaneous Recovery in Dynamical Networks (SRDN) ................. 38
  3.5.19 Autonomous Reconstitution of Compromised Cyber-systems (ARCC) ........................................................................... 39
  3.5.20 Recovery of Infrastructure Networks After Localized Attacks (RINALA) ........................................................................... 39
3.6 Analysis of Recovery Techniques to Date .............................................. 40
3.7 Discussion ...................................................................................... 41
3.8 Summary ....................................................................................... 41

Chapter 4  Automatically Recovering a Network from Persistent OSPFv2 Attacks .... 43
  4.1 Introduction ..................................................................................... 44
  4.2 Background and Prior Work ............................................................. 45
    4.2.1 Existing Recovery Studies .......................................................... 45
  4.3 Fundamental Recovery Mechanisms ................................................. 46
    4.3.1 Restore .................................................................................... 47
    4.3.2 Rollforward ............................................................................. 47
    4.3.3 Reinitialize ............................................................................. 48
    4.3.4 Reconfigure ........................................................................... 48
    4.3.5 Replace .................................................................................. 48
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>OSPFv2 Attacks</td>
<td>19</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Current List of all intrusion response techniques that incorporate recovery.</td>
<td>31</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Current List of other intrusion response techniques that are not reliant on the IDS</td>
<td>32</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Intrusion Response Milestones</td>
<td>40</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Previous security systems utilizing these recovery techniques</td>
<td>47</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Recovery Overhead and Time</td>
<td>55</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Recovery Overhead and Time During DoS Attack</td>
<td>58</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>OSPFv2 Attacks</td>
<td>68</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Overall Packetloss</td>
<td>78</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>Triggered LSA Updates</td>
<td>97</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>Receiving LSA Updates</td>
<td>97</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1 Overview of the five subsystems that make up a resilient architecture . . . . . . 3
Figure 3.1 Diagram of Resilience Architecture designed by Sterbenz et al [Ste10] . . . . . . 22
Figure 3.2 Diagram of Resilience Architecture designed by Smith et al. [Smi11] . . . . . . 24
Figure 3.3 Diagram of Resilience Architecture designed by Goldman et al. [Gol] . . . . . . 25
Figure 4.1 Lab Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
Figure 4.2 Restore demonstrated no recovery . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
Figure 4.3 Rollforward demonstrated no recovery . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
Figure 4.4 Reinitialize demonstrated no recovery . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
Figure 4.5 Reconfigure recovered network in 115s . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
Figure 4.6 Replace recovered network in 35s . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
Figure 4.7 USNET 24Node Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
Figure 4.8 Reconfigure recovered USNET in 122s . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
Figure 4.9 Replace recovered USNET in 42s . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
Figure 4.10 DoS Attack: Reconfigure recovered network in 115s . . . . . . . . . . . . . . . . . . . . 58
Figure 4.11 DoS Attack: Replace recovered network in 35s . . . . . . . . . . . . . . . . . . . . . . . . 59
Figure 4.12 Reconfigure DoS recovery USNET in 122s . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
Figure 4.13 Replace DoS recovery USNET in 42s . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
Figure 5.1 Reference Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
Figure 5.2 VRRC Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
Figure 5.3 Development Network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
Figure 5.4 No Virtual Routers, No program running . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
Figure 5.5 Virtual Routers, without program running . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
Figure 5.6 Virtual Routers, with program running . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
Figure 6.1 NSF-NET-Original . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
Figure 6.2 NSF-NET Link Changes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
Figure 6.3 Software Analysis Heuristic Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
Figure 7.1 Random Time Interval Generator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
Figure 7.2 Lab Topology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
Figure 7.3 Optimization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
Figure A.1 Diagram of LSID Attack. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
Figure A.2 Diagram of MaxAge Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
Figure A.3 Diagram of Max-Seq Number Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
Figure A.4 Diagram of Sequence++ Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
Figure A.5 Diagram of Remote False Adjacency Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
Figure A.6 Diagram of Disguised LSA v.1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
Figure A.7 Diagram of Disguised LSA v.2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
Figure A.8 Diagram of Partition Attack . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
Implementing network recovery is an extremely old, yet under explored part of security research. The vast majority of security research to date has been on preventing the exploitation of security vulnerabilities and detecting these exploitation attempts. Much less attention has been given to the back end of security, which is repairing the damage done by an attack while adapting to prevent further attacks in order to continue operations [Loc08; Zon14; Mer18].

The primary focus of research into routers and routing protocols has been on ensuring fault tolerance, congestion control, load balancing, etc in an effort to make routing protocols as resilient as possible to physical failures (e.g. cut lines, power outages, etc). As a result, modern networks are extremely fault-tolerant, capable of withstanding even catastrophic failures all the while maintaining connectivity. The far-reaching, every-increasing level of connectivity and robustness provided by networks and networked devices has significantly improved the productivity within academia, government, and industry due to the seamless integration of data and operations.

While the distributed operation of routing is robust to traditional Byzantine failures [Per88], it is still vulnerable to deliberate attack. Malicious attacks against the Routing infrastructure do not behave like traditional failures. They intentionally cause routers to fail at a highly accelerated rate, or to propagate incorrect or inconsistent information throughout the network.

The reliance of networks and the devices they support (e.g. laptops, tablets, smartphones, IoT, etc) has significantly increased the vulnerability of organizations, governments, and economies to disruption when the network infrastructure comes under attack [Goe04]. It is within this vast
array of connectivity that the Internet provides that a new threat has emerged. Not from natural
disasters or severed connections, but from malicious actors (script-kiddies, hactivist groups, rogue
nations, etc) who are capable of constantly adapting and devising ever more elaborate means of
compromising data and circumventing network defenses.

When the routing infrastructure itself is compromised, routers have only basic (outdated) meth-
ods of dealing with these situations (e.g. OSPF MD5 Authentication) for confirming that new links
entering the routing table are indeed legitimate. Even if authentication is enabled, at most there is
only a single pre-shared key. These deficiencies in the routing infrastructure can result in various at-
tacks such as: black holing, MiTM attacks, route poisoning, partitioning, etc. Additionally, subverted
control of routing facilitates other types of attacks, such as denial of service, eavesdropping, and
man-in-the-middle. As such, to ensure continued operations, it is necessary that network recovery
be a key component in all future cyber resilience architectures, especially for the OSPFv2 routing
infrastructure.

The routing infrastructure we believe is a high value target, as almost all communication relies
on its correct functioning. There are two levels of routing infrastructure: exterior protocols (e.g,
Border Gateway Protocol (BGP)), which connects roughly 55,000 autonomous systems into the
Internet backbone, and interior protocols (e.g. Intermediate System to Intermediate System (IS-IS)
and Open Shortest Path First (OSPFv2)), which are used by carriers and enterprises to route traffic in
their own network. In this dissertation we have chosen Open Shortest Path First version 2 (OSPFv2)
as the protocol of interest due to the wide-spread adoption and implementation making OSPFv2
global industry standards.

OSPF works in the following way. Routers flood information about their connections throughout
the enterprise network. All routers in the network use this information to form a consistent picture
of these connections, and to choose the shortest path from any point to any point. This consistent
picture, or link state database, depends on receiving timely, accurate, and consistent information
from all the other routers. The protocol refreshes this information frequently, so routing recovers
quickly from ordinary failures such as router and link loss.

1.1 Resilience

In this thesis, we consider system to be resilient if (under normal operating conditions and attacks),
it is capable of:

- maintaining some level of functionality, even when environment changes (e.g. an attack starts)
- improving functionality to full or near-full, when same environment continues (no attack)
- operating without any human intervention or oversight
A resilient network must be capable of not only predicting attacks, incorporating well-known technique, but also take the necessary steps to endure an attack when compromised, while returning to normal operating capabilities. Then automatically update, so as not to succumb to the same vulnerabilities. This multi-layer approach of having multiple systems capable of working in conjunction as well as independently of one another will make the network infrastructure vastly more resilient to malicious actors.

**Figure 1.1** Overview of the five subsystems that make up a resilient architecture

A brief summary of each of the five resilience techniques are as follows:

- **Anticipate**: Ability to utilize data in such a way as to forecast the next attack.

- **Prevent**: Measures taken to actively look for vulnerabilities within ones one system to secure weaknesses before they are exploited.

- **Detect**: Typically a reactive measure, systems designed to look for abnormalities or signatures of known attacks.

- **Recover**: Designed to function in a post attack environment with the intent of returning resources to a robust state.
- Adapt: Update system defenses so they will no longer be vulnerable to same type of attack or exploit.

The primary means of network security, to date, has relied on basic intrusion prevention and intrusion detection. Intrusion prevention is both an active and passive phase in security. In prevention, the overall goal is to actively take steps to prevent an attack. For example, by hiring white hats to actively determine weaknesses in an organization's network or adding network flow encryption to prevent a man-in-the-middle attack. The detection phase of security can also be both active and passive. Detection scans network traffic looking for both attack signatures and suspicious activity. The detection phase is where firewalls and Intrusion Detection Systems (IDS) are located.

Intrusion Response Systems (IRS) are part of the IDS. Modern IRS' are designed to choose the appropriate responsive action to an attack based on the alerts received from the Intrusion Detection System (IDS) and from there take evasive/corrective actions to ensure safety [Sta07]. In other words, while not all responses to the detection of an attack may be to recover, the choice to ‘recover’ is always a ‘response’.

Recovery is designed to function in a post attack environment with the intent of returning cyber resources to a robust state. Recovery has been placed at various stages within the security process. In most cases recovery is considered to be an independent technique, while in other cases, it is considered to be an accessory of the IRS. Therefore, it is important to reiterate that the IRS is not the same thing as recovery. Intrusion response is predicated on the ability to detect one’s adversary, while recovery is what is necessary in the aftermath of a successful attack. Adapt is the final system utilized in a resilient architecture. During the adaptive stage, the networked systems are designed to adapt to the effects of an attack to keep an attack from being effective. In the best case scenario, a network will be able to adapt before the attack occurs, while in the worst case, a network will have to adapt as a result of recovery in the aftermath of an attack.

Each subsystem within the resilient architecture is designed to work cooperatively and independently of one another. The architecture depicted in Figure 1.1 acts as a visual aid for the following examples: If a OSPFv2 network incorporating a resilient architecture with the above five subsystems is able to anticipate an attack, then the system will be able to immediately adapt in order to keep the attack from exploiting or infiltrating the infrastructure (An -> Ad). Another example could be a scenario where the attack was not anticipated or prevented. In this scenario, once the attack was detected, a recovery system would be triggered to restore the network to a functioning state in which the attack no longer exists, while simultaneously adapting so as to no longer be vulnerable to the same attack (D -> R -> Ad).

The purpose of this dissertation is to motivate the need for making the OSPFv2 routing infrastructure more resilient to modern attacks. Specifically, preventing spoofed LSAs, detecting persistent OSPF attacks, and finally triggering automated-recovery.
1.2 Thesis Statement

As previously mentioned, while the routing infrastructure is extremely resilient towards natural failures, this ceases to be true when malicious intent is the source of failure. Security researchers must look beyond the conventional status quo of traditional ‘detection’ and ‘prevention’ techniques to address this problem. Therefore in order to maintain the level of connectivity that people have come to expect, a new means of recovering from these malicious attacks must be developed. The thesis statement of this work is the following:

OSPFv2 can be made more resilient to persistent OSPFv2 attacks through the introduction of LSA inspection, Layer 2 data, OSPFv2 router implementation, and LSA timing.

In this dissertation, the primary focus of our investigation is on making OSPFv2 more resilient to malicious actors by evaluating the success of various techniques we have developed to prevent, detect, and recover.

While there are other Layer 3 routing protocols such as the Routing Information Protocol (RIPv2 and the Cisco proprietary Enhanced Interior Gateway Routing Protocol (EIGRP), they remain outside the focus of our research. We also do not attempt to address network attacks other than those that specifically target OSPFv2.

1.3 Contributions

In this dissertation, we make the following contributions:

- We analyze the effectiveness of the five accepted recovery techniques (Restore, Rollforward, Reinitialize, Reconfigure, and Replace) against persist OSPFv2 attacks. The routing infrastructure of the Internet is a high-value target for malicious actors. The OSPFv2 protocol is one of the most widely used protocols for Autonomous System (AS) internal routing, and has been the subject of attacks targeted at partitioning the network. Because of the widespread existing penetration of OSPF, research in defending against such attacks has focused toward prevention and detection techniques working in tandem with OSPFv2 routers, rather than attempted modification of the protocol itself. However, comparatively few studies have attempted to address the topic of a network capable of recovery, which signifies a system that can be proof to an attack without the necessity for any specific attack to be explicitly detected, localized, or countered. We first examine these existing recovery techniques and come up with the underlying common mechanisms that they utilize. Then we examine the continued usefulness of these approaches with the more modern persistent OSPFv2 attacks, by running actual attacks...
against an isolated network formed of typical commercial network elements, while allowing each such recovery mechanism to defend the network. Our experiments show that three of the five basic mechanisms can no longer defend against Partitioning or DDoS attacks when attacks are persistent, and provides relevant performance results for the other two. Our results also point the way to further improving these mechanisms for even more sophisticated attacks, in the future [Mer18].

- **We analyze the effectiveness of the Virtual Router Resiliency Cluster (VRRC) in detecting and recovering the OSPFv2 routing infrastructure from persistent OSPFv2 attacks.** The VRRC was designed to add resilience to the OSPFv2 routing infrastructure by providing an upgradeable platform on which multiple recovery ‘modules’ (characteristics) can be implemented. In its current form the VRRC employs three base techniques. For convenience the techniques are labeled as Deep LSA Inspection (DLSAI), Multi-Layer Authentication (MLA), and Forced Routing Updates (FRU). DLSAI utilizes real-time, deep LSA sampling, through an ever-increasing set of filters to narrow down the possible sources of the attack. Concurrently, MLA checks against data gathered from Link Layer Discovery Protocol and Cisco Discovery Protocol (with Simple Network Management Protocol used for devices multiple hops away) allowing the network to remain dynamic while comparing Layer 2 data against information gathered from deep-LSA sampling. Finally, the initial recovery module, FRU is designed to purge the routing tables in the event a compromised link is detected. While simultaneously randomly electing a new Designated Router and Backup Designated Router. This chapter explores the initial success of these three techniques in an effort to build a next-generation IRS [Mer19].

- **We establish that implementation differences observed at the packet level between physical OSPFv2 routers can be used to differentiate between routers.** The widespread adoption of OSPFv2 as the interior gateway routing protocol of choice within enterprise and some service provider networks could make the protocol an attractive target for bad actors. While OSPFv2 is an open standards based protocol, the specification is non-exhaustive leaving some design decisions to the implementer. Bad actors may target OSPFv2 deployments by masquerading as routers, performing router-in-the-middle or other similar techniques. The ability to detect when one router is falsely claiming to be another router through the analysis of the routers fingerprint would be a powerful tool in the network operators arsenal. To date there has been little research on fingerprinting OSPFv2. We analyze multiple packet captures taken from both physical and virtual instances of Cisco and Juniper routers to confirm the presence of differences in OSPFv2 implementations and to characterize these differences. From this analysis we describe heuristics for predicting the OSPFv2 router based on identified characteristics and develop a software utility to conduct this analysis on packet captures. Ultimately we hope this analysis will provide valuable information enabling network operators to improve network
security posture by making it more difficult for bad actors to launch attacks against OSPFv2 networks.

- We analyze the effect of the LSA Synchronized Time Controller (LSTC) in preventing spoofed LSA Updates. Routing protocols are essential for correct network function and therefore make attractive targets for attackers. The OSPFv2 protocol is one of the most widely used protocols for Autonomous Systems (AS) internal routing. OSPF has long been the subject of attacks designed to eavesdrop, partition, black-hole, or man-in-the-middle the network. Our proposed solution adds the LSA Synchronized Time Controller (LSTC) to the network infrastructure. The LSTC makes use of a deterministically pseudorandom number generator designed to calculate time intervals for LSA Updates, thus randomizing the LSA update times on all routers and increasing the difficulty in gleaning the necessary information to launch a targeted network attack. We show that the LSTC makes the network more resilient against all known persistent OSPF attacks and that the concepts are feasible to implement without resorting to the complexity of key management, and without modifying the routing protocol.

1.4 Dissertation Outline

The goal of this dissertation is to investigate practical solutions for triggering auto-recovery to ultimately make the OSPFv2 routing protocol more resilient.

In Chapter 2, we present a summary of TCP/IP, OSPFv2, and routing. We further go into detail by presenting the vulnerabilities of OSPFv2 against specific attacks.

Following the background, we discuss the related work in Chapter 3 by presenting a summary that focuses on the terminology and related work in the field of recovery.

Chapter 4 explores the current recovery techniques: Restore, Rollforward, Reinitialize, Reconfigure, and Replace. Once these recovery techniques are reviewed, their success in being able to recover from a modern persistent OSPFv2 attacks is evaluated.

In Chapter 5 we further provide a synthesis between a resilient architecture by incorporating a rudimentary recovery mechanism that is triggered by a three-tier detection method to form the Virtual Router Redundancy Cluster.

Chapter 6 addresses resilience from the perspective of detection. By identifying the differences in OSPFv2 router configurations parameters that are observable within packet captures, we take the first steps in demonstrating that it is possible to detect with malicious actors have spoofed legitimate routers.

Chapter 7 continues to build upon our previous research by providing even more information for a auto-recovery system regarding when to initialize recovery. This is done in two parts: First, we demonstrate that it is possible to make modern routers in distribution generate LSAs at specific
instances, overriding their periodic nature, without extensive modification. Secondly, we demonstrate that with such a synchronized pseudorandom LSA distribution, we can nullify attacks against OSPFv2 that depend on sending spoofed LSAs.

Finally, Chapter 8 provides direction for future research into recovery. We reiterate the need for ‘recovery’ as a key component in a resilient OSPFv2 network.
The purpose of this chapter is to provide the reader with a summary of the operations of the OSPFv2 routing protocol and all the known attacks against OSPFv2. In Section 2.2 we begin by providing the reader with a detailed description of the workings of OSPFv2. Section 2.3 reviews how the attacks are able to exploit OSPFv2 and other routing protocols. Then in Section 2.4 we review in detail the attacks specific to OSPFv2.

2.1 TCP/IP and the Routing Infrastructure

The TCP/IP Routing Infrastructure is the foundation for modern routing protocols (e.g. BGP is TCP based and OSPFv2 is IP based). The TCP/IP routing infrastructure is a four-layer model consisting of the: Application, Transport, Internet, Link layers. The scope of this research focuses on the TCP/IP Internet layer, or more commonly known as the network layer in the OSI model. Each layer of the TCP/IP model has several protocols that provide a specific functionality [RM88].

The network layer is responsible for routing, addressing, and subnetting this is the layer routers operate in. For routers to effectively and reliably send data over the network they rely on routing protocols, OSPFv2, IS-IS, and RIP to name a few. The job of the network layer is to forward packets across multiple hops across any nodes that are connected to each other.
2.2 OSPFv2

Open Shortest Path First (OSPFv2) is an “Open standard” classless Interior Gateway Protocol (IGP) that uses the shortest path first “Dijkstra’s algorithm” to calculate the shortest path to each destination. OSPFv2 incorporates a two-layer hierarchy whose primary elements are the area and the autonomous system (AS). The area is a grouping of contiguous networks (logical subdivisions of the AS), while the AS consists of a collection of networks sharing a common routing strategy. The AS can be logically subdivided into multiple areas under a common administrator [Moy98a].

2.2.1 OSPFv2 Operations

OSPFv2 is also a link state routing protocol. A link state routing protocol is best summarized by describing the link as an interface on router and the state “State of the Link” as both the description of that interface (e.g. interface IP address, subnet mask, type of network to which it is connected, routers that are connected to network) and its relationship to neighboring routers. OSPFv2 establishes these neighboring relationships through the exchange of ‘hello’ packets. OSPFv2 then propagates these Link State Advertisements (LSAs) rather than exchanging routing table updates. During the update process, the LSAs are flooded to all OSPFv2 routers in the area (they do not have to be directly connected). The OSPFv2 database is pieced together from LSAs generated by the OSPFv2 routers. Routers periodically send out their LSA updates (by default every 30min or immediately if the routing state changes) to advertise their current state. It is the collection of all the link-states that form the link state database.

The LSA process for updating a routers database when multiple instances of the same LSA are received, as stated in [Jef05] that, “to determine which LSA is the most recent, perform the following calculation:

1. Compare the sequence number. The LSA with the highest sequence number is more recent
2. If the sequence numbers are equal, then compare the checksums. The LSA with the highest unsigned checksum is the more recent.
3. If the checksums are equal, then compare the age. If only one of the LSAs has an age of MaxAge (3600 seconds), it is considered the more recent. Else:
4. If the ages of the LSAs differ by more than 15min (Known as the MaxAgeDiff), the LSA with the lower age is more recent
5. If none of the preceding conditions are met, the two LSAs are considered identical”
2.2.2 Strengths/Weaknesses

OSPFv2 has several components (flooding, fight-back, link authentication, and hierarchical routing) that make it a very robust and resilient routing protocol against traditional network failures. Unfortunately, flooding, fight-back, etc is not sufficient when it comes to a malicious attacks [Jon] [Wu99] [Moy98a].

- Flooding: OSPFv2’s flooding algorithm is very reliable. The flooding mechanism is the means by which OSPFv2 propagates its LSA throughout the local network to ensure that all routers in the same area have the same topological database.

- Fight-Back: When a router running OSPFv2 receives an instance of its own LSA, more recent than the last one it originated, it will advertise a newer instance of the LSA. The router will either flush or update the false LSA, thus ensuring consistency with all the links.

- Link Authentication: If enabled, OSPFv2 is capable of authenticating every packet sent across the network using a secret shared key. However, this key must be manually configured on every router, so it is often the case that all routers in the AS use the same key if authentication is even enabled at all.

- Hierarchical Routing: Hierarchical routing was designed to deal with scalability, reduction in routing table size, reduction in routing resources, etc.

2.3 Routing Attacks

Routing protocols by their very design are resilient, capable of adapting to and recovering from ‘natural’ (disasters, cut wires, power failure, etc) and traditional Byzantine faults (accidental misconfigurations and faulty equipment). However, the traditional robust and resilient nature of these routing protocols was never designed to endure concentrated and persistent targeted attacks (e.g. attacks that exploit the implementation weaknesses of these routing protocols). The rest of this section is designed to further motivate the need for a recovery system by addressing all possible “Damages” that will result from a routing attack, as accepted by the Network Working Group and the IETF.

**Threat Action/Source**: realization of an attack (when a system comes under attack). The Threat Actions/Source \(^1\) against routing protocols may be broken down into two categories [RFC 4949, 4593] [Shi07][Bab06]:

- Outsider Attacks: These attacks may come from anywhere outside the local network and have ability to send IP traffic to a router, possibly observe the routers’ replies, and potentially control

\(^1\)Moving forward, Threat Source will be the primary term used
the path for a legitimate peer's traffic (outsiders are not considered legitimate participants in routing protocol).

- Byzantine: These are considered legitimate participants in the routing protocol and as a result are capable of subverting the router (these can also be the result of misconfigured or faulty routers). R. Perlman states that the problem with Byzantine failure is that, “a node with a Byzantine failure may corrupt messages, forge messages, delay messages, or send conflicting messages to different nodes” [Per88]. As a result a compromised router (just as with a misconfigured or faulty router) is capable of transmitting bogus information, because the ‘compromised’ router is considered a ‘legitimate’ peer/neighbor.

**Threat Consequence:** the resulting actions of an attack, as defined in RFC 2828 and 4949 [Shi00][Shi07]. The Threat Consequences against routing protocols are broken down into four general categories:

- Disclosure: attacker successfully accesses routing information
- Deception: legitimate router receives a forged routing message
- Disruption: legitimate router operations are interrupted or prevented
- Usurpation: attacker gain control over service/function of legitimate router

Now that we have classified the source and subsequent consequences of an attack against the routing protocols, we will list the damage that might result from an attack. The damage is classified as the Threat Consequence Scope. The Threat Consequence Scope (TCS) can be either damage done against the routing infrastructure as a whole (Damage List 1 [Bab06]), or damage that might result in an attack against a individual host/network address [Bab06]. However, for the purposes of this proposal we will only focus on the damage against the routing infrastructure.
Damage List 1.

- **Network congestion**: More data traffic is forwarded through some portion of the network than would otherwise need to carry the traffic.

- **Partitioning**: Some portion of the network believes that it is partitioned from the rest of the network when it is not.

- **Black hole**: Large amounts of traffic are unnecessarily re-directed to be forwarded through one router and that router drops many/most/all packets.

- **Churn**: The forwarding in the network changes (unnecessarily) at a rapid pace, resulting in large variations in data delivery patterns (adversely affecting congestion control techniques).

- **Instability**: The protocol becomes unstable so that convergence on a global forwarding state is not achieved.

- **Clog**: A router receives an excessive number of routing protocol messages, causing it to exhaust some resource (e.g., memory, CPU, battery).

- **Overcontrol**: The routing protocol messages themselves become a significant portion of the traffic the network carries.

- **Looping**: Data traffic is forwarded along a route that loops, so that the data is never delivered (resulting in network congestion).

These examples of attacks against OSPFv2, IS-IS, and BGP demonstrate how traditional methods fail to prevent, detect, or recover from an attack launched by an intelligent adversary.

- Outsiders: Firewalls and IDS can be compromised or circumvented

- Byzantine Robustness: Will also break down in the case of an intelligent adversary [Ram13] [Pau07]

The three attacks chosen to demonstrate how traditional methods fail are the Partition, Replay, and Route Injection attacks. For convenience, we systematically break each attack down into its respective category based on: Threat Source, Threat Consequence, and Threat Consequence Scope.

1. **OSPFv2**:
   
   (a) **Source**: The partition attack [CN15]

   i. **Consequence**: The Partition attack would result in both Deception and Usurpation and could be committed by an Outsider or Byzantine.

---

2An attack against BGP was included to emphasize that this is not just a problem for Link-State routing protocols.
A. Consequence Scope: Partitioning, looping

2. IS-IS:

   (a) Source: The Replay Attack [HK12]

      i. Consequence: The Replay attack would result in Disruption of legitimate routing operations

      A. Consequence Scope: Churn, Instability

3. BGP:

   (a) Source: Route Message Insertion/Deletion [Bab06][Mur06]

      i. Consequence: Message Insertion/Deletion would result in Deception and Disruption to routing operations

      A. Consequence Scope: Starvation, Black hole

The current deployment of OSPFv2 and IS-IS were standardized well over a decade ago and are the primary ‘interior’ routing protocols in use today, and will continue to be for some time (this is primarily due to the fact that implementing a new routing protocol takes decades to standardize).

While the slow-adoption of upgrading and updating routing protocols is advantageous for attackers, this same weakness can also be made advantageous for developing a recovery system. The Threat Consequence Scope clearly defines/identifies all possible ways for the routing protocols to be exploited/corrupted as they exist in their current release. This is bolstered even further by the knowledge that routing attacks in the late 1990’s, resulted in the same damage as the routing attacks from 2016. The only difference is the Threat Action. The Threat Consequences and the Threat Consequences Scope remain the same [Bab06][Mur06][Moy98a] [Cal90][Shi07][Sri16].

2.4 Specific Attacks Against OSPFv2

OSPFv2 is vulnerable to a myriad of generic attacks such as: eavesdropping, message replay, message inserting, message deletion, message modification, Man-in-the-Middle, and Denial-of-Service. The following attacks fall into two categories, non-persistent and persistent, which describe whether or not the attack is able to avoid the fight-back mechanism. While we provide the reader with a fairly comprehensive review of the attacks against OSPFv2, the four primary attacks that are the focus of this research are: Remote False Adjacency, the Disguised LSA v.1 (D-LSA v.1), Disguised LSA v.2 (D-LSA v.2), and the Partition Attack.
2.4.1 OSPFv2 Vulnerabilities

[Jon] provides a comprehensive analysis of OSPFv2 vulnerabilities. One such example is falsifying ‘HELLO’ messages, resulting in a ‘forced’ DR election, while another is using ‘HELLO’ messages to flood the victim router (essentially a DoS attack). However, the most novel of these findings were attacks capable of subverting/disabling the fight-back mechanism of OSPFv2. Periodic Injection and Phantom Routers are two examples of attacks capable of subverting the OSPFv2 fight-back. As stated in RFC 2328, OSPFv2 will update no faster than “once every 5 MinLSInterval.” Therefore, if an attack floods at a rate higher than once every MinLSInternal, the attacker will be able to directly affect the routing domain. All of this can occur because the malicious LSA is considered newer than the legitimate LSA (*see LSA calculation check above). In the case of the phantom router, information injected in the routing domain, from false (phantom) routers, will not trigger a fight back. These false-routers will remain in the link state database until the MaxAge expires (1hr default), however these false links do not propagate thorough out the network.

2.4.2 Table Overflow Attack

[Wan97] was one of the first to address potential LSA attacks. One such example, The Table Overflow Attack, is designed to impersonate the router that resides on the border of the AS, while simultaneously advertising links to destinations outside the AS. Additionally, it is the only LSA that is propagated throughout the entire AS (save the stub areas). The consequences of this attack result in the attacker being able to eavesdrop or black-hole traffic. One of the reasons for the success of this attack is because OSPFv2 lacks the ability to authenticate the roles that external routers assume.

2.4.3 LSID

[Jou00] Also describes the LSID attack (*Figure A.1*). Here the attacker intercepts the LSA and then sends back a modified version. This modified LSA is set to have the Link State ID != to the Router ID, the index pointer to database equal to null. The pointer is then used to access the ls_age, which will create a segmentation fault. Jou et al. details several more attacks (*Seq++, MaxAge, MaxSeq# Attack*) against OSPFv2, however, all four attacks listed will trigger the fight-back mechanism in OSPFv2. Additionally, these attacks are considered ‘non-persistent’ and will result in the attacker having to re-launch their attack.

2.4.4 Max-Age

The Max-Age attack is designed to set the age field of an LSA to ‘MaxAge’ thus causing the valid LSA to be flushed from all the routers reached by the flooding mechanism. The owner of the LSA will fight back by issuing a new LSA with age set to 0 and a higher sequence number. Any attack exploiting
this vulnerability could cause unnecessary flooding and refreshing of the Link State Database, hence making the routing information inconsistent. Routers that do not have a copy of the LSA in their Link State Databases will not contribute to the flushing of it, this can help the owner of the LSA in its fight back [Jou00] [Jon].

2.4.5 Max-Seq Number

The Max-Seq Number is generally considered to be more of an implementation, rather than a protocol vulnerability. Nonetheless it is listed in this memo for historical reasons and because at least one recent implementation of OSPFv2 was still affected by it. This bug concerns LSA sequence number roll-over. When an LSA sequence number reaches its maximum value (0x7FFFFFFF) it is not flushed by flooding it with its age set to MaxAge; instead, the erroneous implementation will simply reissue the LSA with a rolled-over sequence number (0x80000000). Any new LSA instance will always be considered outdated when compared to previous sequence number (which is set to max-value). Thus, an attacker could utilize this bug to subvert the fight back mechanism and install a bogus LSA on all routers for a MaxAge-long interval without any effective fight back from the owner of the LSA [Jou00] [Jon].

2.4.6 Seq++

The Seq++ attack is the result of an attacker incrementally modifying the link-state. Once the attacker receives an LSA instance, the attacker then modifies the link state metric and increase the LSA sequence number by 1 (i.e. Seq++). The attacker also needs to re-compute both the LSA and OSPFv2 checksums before the tampered LSA instance is re-injected into the system. This attacking LSA, because it has a higher sequence number, will be considered “newer” by other routers. Eventually it will be propagated to the victim router of which the original LSA belongs to. The victim router, according to the OSPFv2 specifications, will “fight back” with a new LSA carrying correct link status information and an even fresher sequence number. The effect of this attack an unstable network topology if the attacker keeps generating “Seq++” LSA instances. “For example, all routers at one point will think the link cost is big (e.g. 100) but then the fight-back LSA instance from the originator will tell them the cost metric is much smaller (e.g. 2)” [Jou00] [Jon].

2.4.7 Remote False Adjacency

The Remote False Adjacency (RFA) attack (also known as the Periodic Injection Attack presented by [Jon]) (Figure A.5) is when the attacker impersonates a phantom router located on the local network of the victim. The victim router then advertises on behalf of the local network, an LSA containing a link to the phantom router. The result/consequence of this attack is that the attacker can confuse
the victim into establishing adjacencies to a phantom route on the victims local network, essentially black-holing the router [Nak12].

2.4.8 Disguised LSA v.1

The D-LSA v.1 attack (Figure A.6) takes advantage of the exploit in how OSPFv2 calculates LSA. According to OSPFv2, LSA are “considered Identical if they have the same Seq. #, Checksum, and Age.” The D-LSA v.1 attack exploits this by advertising a disguised LSA that matches a recently generated LSA, that has yet to be installed by all routers in the AS. The goal being to send a disguised LSA and a valid LSA, so the disguised LSA arrives at victim immediately before the valid instance, resulting in the disguised LSA being installed in the routing table; the result of which is route poisoning. However there are several drawbacks such as: not all AS are poisoned, attacker required to send two false LSAs, and no effect on routing table [Nak12].

2.4.9 Disguised LSA v.2

The D-LSA v.2 attack (Figure A.7) is a variant of the original D-LSA v.1 attack and does not suffer from the three drawbacks of the original attack. After the attack, both the false and the legitimate LSA reside in the LSA database of all routers. However, when a router calculates what links should reside in its database, it only uses the Link State ID field, as opposed to all three fields (LS Type, Advertising Router, and LS-ID). Nakibly et al. observed that due to the ambiguity in the wording regarding how OSPFv2 forms their LSA databases, it was possible for both LSA to reside in the database simultaneously. As such, the way the router chooses the appropriate LSA to install into its database must be “implementation/vendor” dependent. The consequences of this attack are valid LSAs replaced with false LSAs, routing tables of all routers poisoned (save the victim), and the routing table of the victim is erased [Nak14].

2.4.10 Partition Attack

That Partition Attack (Figure A.8) is the latest in attacks against OSPFv2. The new attack is designed to partition the network into two parts. This is accomplished by having the compromised router send out different LSAs, on different ports, with the same header information in an attempt to have each part create a different routing map. However, one of the greatest strengths of this attack is that it works even if the LSAs are digitally signed. The author states that, “Judicious use of “self-LSA” falsification attacks can overload remote links and routers that are far from the attacker, while leaving no obvious data plane traces that may lead to the attacker.” The Partition Attack can result in forwarding loops, lengthened routing paths, and disconnected routers. Additionally, is slightly more difficult to recover from because the routers no longer see a path to one-another [CN15].
2.4.11 Loki

Loki is a primarily layer-3 network attack system written in python and C and built around a GTK-based GUI, a back-end networking API, and a series of vulnerability-exploiting ‘modules’ constructed using the networking API. One of the modules Loki contains is centered on the OSPFv2 protocol. This OSPFv2 attack module implements both OSPFv2 route injection and OSPFv2 authentication cracking attacks. The route injection attack works by establishing a ‘full’ OSPFv2 neighbor relationship with a real, authenticated router. By following the full neighbor establishment process while posing as a new, physical router, an attacker can gain a trusted position on a target network. This position in the network allows the attacker to both access the router’s full OSPF table (giving them a detailed map of the entire network), and also to add himself or another machine to the routing table, resulting in a Man-In-The-Middle attack. Additionally, an attacker could add non-existent routes to the table, potentially black-holing traffic destined for other locations on the network [Ren10].

Many attacks (such as ospf-ash.pl and several Scapy scripts) which came about as a result of the wide availability of networking frameworks which were collected and re-implemented in the “All your packets” toolkit; the direct precursor to Loki, in 2009. Most attacks present in Loki existed prior to its development, and they were combined with other attacks or extended beyond their original functionality. For example, MD5 cracking tools were combined with an OSPFv2 route injection attack to allow for route injection even in an MD5-authenticated OSPFv2 network [Ren10].

2.5 Brief Summary of Proposed Countermeasures

Several studies have been conducted on resilient networks, [Smi11] however most of this research does not address malicious threats. Additionally, most solutions involve either making significant changes to the network infrastructure or to the protocols themselves. In the case of attacks against OSPFv2, specifically exploits against LSA, several solutions were proposed: [WW98] suggests the use of a secure wrapper with different levels of checking gates for input and output between the secure wrapper and the protected router. The secure wrapper examines all data, using a security policy database to determine the status of these checking gates [Wu99]. introduces a scalable IDS incorporates a ‘detection’ module that performs statistical and protocol-based intrusion analysis and allows the user to remotely modify the ‘prevention’ and ‘detection’ modules based on the detection information. [Nak12] discusses two potential measures that could be taken to prevent the remote false adjacency and disguised LSA. The remote false adjacency could be thwarted if the AS ensures that different links use independent secret keys, however this technique would be impractical because the key management and implementation would have to be done manually. In the case of the D-LSA v.1 attack, it was proposed that one could mitigate the predictability of the LSAs by advertising false links with random values to randomize the LSA checksum. Unfortunately,
this technique would also be impractical for deployment in real-world applications because it would ultimately result in increased routing table sizes.

*Table 2.1* is included to provide the reader with a summary of the aforementioned attacks. For convince, the categories are:

- **Persistent**: is the attack natively persistent.
- **Patched**: Has the attack been patched by the vendors ³
- **Encryption**: Would enabling OSPFv2 MD5 encryption prevent attack from ‘untrusted’ routers ⁴
- **Proposed Technique**: reference to proposed solutions

---

**Table 2.1 OSPFv2 Attacks**

<table>
<thead>
<tr>
<th>Attack</th>
<th>Persistent</th>
<th>Patched</th>
<th>Encryption</th>
<th>Proposed Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxAge</td>
<td>No</td>
<td>Yes ³</td>
<td>Yes</td>
<td>[Jou00]</td>
</tr>
<tr>
<td>Seq++</td>
<td>No</td>
<td>Yes ³</td>
<td>Yes</td>
<td>[Jou00]</td>
</tr>
<tr>
<td>MaxSeqNum</td>
<td>Can be</td>
<td>Yes ³</td>
<td>Yes</td>
<td>[Jou00]</td>
</tr>
<tr>
<td>Remote False Adjacency</td>
<td>Yes</td>
<td>Yes-[JSA16]</td>
<td>Yes</td>
<td>[Nak12]</td>
</tr>
<tr>
<td>D-LSA v.1</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>[Nak12]</td>
</tr>
<tr>
<td>D-LSA v.2</td>
<td>Yes</td>
<td>Yes-[CSA13][JSA13]</td>
<td>Yes</td>
<td>[Nak14]</td>
</tr>
<tr>
<td>Partition Attack</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[CN15]</td>
</tr>
</tbody>
</table>

---

³Verified in lab through testing against recent copies of Junos and IOS
⁴Encryption will prevent these attacks from succeeding as long as none of the trusted routers have been subverted

This chapter is designed to provide the reader with an overview and a brief background of the primary routing protocol (OSPFv2) and its attacks that referenced throughout this dissertation. In the next chapter, we discuss the current state of network recovery.
In this chapter, we provide a brief overview of several resilience architectures before providing a detailed literature review of security systems that have incorporated some form of recovery. Section 3.2 provides the reader with a review of several proposed resilience architectures, all depicting recovery as a key component. Next in Section 3.4 we provide a list of terminology. Section 3.5 is designed to provide the reader with a comprehensive summary of all related work that pertains to recovery. The rest of the chapter goes on to discuss and finally conclude the related work.

3.1 Revised Network Resilience Architecture

There is a tremendous need for recovery, especially a means of automating the process that triggers recovery, in the event traditional methods do not detect the attack. Designing such a recovery system has been an ongoing problem for researchers spanning several decades. While a number of innovative defensive [Per88] and response [Zon14] techniques have been proposed, they still conform to traditional cyber “resilience” models (e.g. CMU-Computer Emergency Response Team (CERT), Alliance for Telecommunications Industry Solutions, Alliance for Telecommunications Industry Solutions (ATIS), and the Advanced Networked Systems Architecture (ANSA) [Ell97] [Zol94] [Zha93].

Since 2008-2009, there has been a tremendous surge in refining and redesigning “cyber” resilience architectures [NS11] [Bes08]. These next-generation resilience architectures are designed
with the ideology of adversarial success. This change in mindset from the traditional, ‘prevention and detection’ status quo, has resulted in new resilience architectures being designed to not only incorporate the traditional fields of security, but also have the capability of minimizing damage, enduring (fighting-through) the attack, and ultimately recovering from any damage done [Kha] [Ste10] [Gol11]. The first section of this chapter reviews several different next-generation resilience architectures. Next, section 3.2.4 briefly compares these resilience architectures and their approaches for incorporating recovery. Finally, section 3.3 proposes our resilience architecture which combines the very best of the reviewed resilience techniques into a multi-layer resilience architecture with recovery as one of the key components.

3.2 Review of Resilience Architectures

3.2.1 Cyber Resilience Engineering Framework (CREF)

Khan et al. define resilience as “the ability of a system to resist against (or minimize) the potential damage in order to maintain system state within an accepted operational level in response to any external threat or attack.” Using this as the basis for what a resilient system should be, Khan et al. developed CREF. CREF is designed to increase the resilience of a network incrementally by refining the system after faults have occurred. In doing so, it must also measure the resiliency with respect to various attacks and properties. CREF is a generic framework used to measure cyber resilience at a multitude of levels and perspectives. The framework accomplishes this through the use of construction, which assumes that initially any system is vulnerable and not resilient. Using algebraic Petri nets and temporal descriptions to build a model, counter examples can be found and are used to refine the model.

The measurement of resiliency relies on many metrics, and the authors categorized them into three groups: pro-active, resistive, and reactive. A network can be pro-actively resilient if attacks do no damage to the network. Resistive resilience refers to the ability of a network to resist an ongoing attack. When damage has already been done to the network, reactive resilience determines whether the network can return to a functional state in the aftermath; CREF’s framework intends to encompass all of these metrics (Figure 3.1. CREF has several important concepts that are used as inputs to produce a resilience axis for the network. These concepts are: network model/program, attacks/properties, and resilience metrics. Network model/program refers to the representation of any logical or physical network system. Attacks/properties refers to the broad characterization of resilience of a network. Resilience metrics were previously defined. In order to maintain a generic framework, CREF allows any resilience metrics to be input, not just specific ones. CREF then determines a measure of resilience based on each input and determines a global resiliency for the entire network. The authors use a computer firewall to demonstrate creating a resilient network through
construction. They measure the resilience of their simulated network using CREF and the attack they use for the network is a denial of service attack. In the most basic version of the network, the firewall rule policy allows default access for any packet. Since this is easily exploited by attackers attempting to overload the firewall, this system is not resilient to attacks. Once CREF has used model checking for the measurement of resilience, the denial of service attack is modeled to a temporal description. When model checking the property and learning that it is not satisfied, the firewall can be redefined to redirect bad packets. This model checking of a property and redefining the firewall is done until the property is satisfied and the network is measured as resilient. While their case study was only an initial version of a framework, it provides a solid foundation for resiliency of a network at multiple levels [Kha].

![Figure 3.1 Diagram of Resilience Architecture designed by Sterbenz et al [Ste10].](image)

### 3.2.2 Resili-Net

Sterbenz et al. define resilience as the “ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation.” There are a number of different disciplines that serve as a basis of network resilience, most of which have been developed independently over time leading to no consistent scheme and terminology. Sternbenz et al propose an organization of the disciplines grouping them into “challenge tolerance” or “trustworthiness.” The challenge tolerance disciplines deal with designs and engineering of systems that continue to provide service in the face of challenges, looking more closely into fault tolerance and survivability,
disruption tolerance, and traffic tolerance. The trustworthiness disciplines help to describe the properties of resilient systems, which include dependability, security, and performance.

The resilient and survivable networking initiative (ResiliNet) is an initiative that comes out of the University of Kansas which aims to understand and progress the state of resilience and survivability of computer networks. The ResiliNet framework was heavily influenced by previous frameworks designed by ANSA, T1A1.2 working group of Alliance for Telecommunications Industry Solutions (ATIS), CMU-CERT Coordination center, and the Survivable Mobile Wireless Networking (SUMON-WIN) project.

The base of the ResiliNets strategy includes four basic principles: faults are inevitable and it is not possible to construct a fault free system, an understanding of a normal operation is necessary, it is necessary to expect and prepare for adverse events and conditions, and in order for a network to be resilient a response to an adverse event and condition is required. ResiliNets’ strategy for resilience consists of two phases. The first phase is a passive core with a cycle of 4 steps that are performed in real time. These 4 steps are: defend against challenges and threats to normal operations, detect when an adverse event or condition has occurred, remediate the effects of the adverse event or condition, and recover to original and normal operations. The second phase consists of diagnosing the fault that was the cause of the error and refine the actions from phase 1 to increase the resilience of the network (Figure 3.2).

By taking into account the four principles and the two phase strategy, Sterbenz et al constructed a list of resilience principles that could help design resilient networks and systems. The principles are clustered into 4 major categories: prerequisites, tradeoffs, enablers, and behavior [Ste10].

While the scope of this proposal is ‘recovery’, it is important to note that incorporating ‘recovery’ as a key component in cyber resilience architecture has been proposed in several well-known next-generation cyber resilience architectures. In the next section, we will compare these resilience architectures and specifically discuss how and why ‘Recovery’ plays such an important role in resilience.

3.2.3 $D^2R^2 + DR$

Smith et al. define resilience as the ability of a network to defend against and maintain an acceptable level of service in the presence of such challenges (e.g. malicious attacks, software/hardware faults, human mistakes, and natural disasters). The resiliency architecture of $D^2R^2 + DR$ (Defend, Detect, Remediate, Recover, and Diagnose and Refine), builds upon the work of Sterbenz et al. Defend, Detect, Remediate, Recover, and Diagnose and Refine. A real-time control loop was used to improve the design of the network based on past operations. The authors implement network resilience using the aforementioned strategy by deriving resilience metrics, distributed information store, and policy-based management. Smit et al. developed a resilience control loop in which “a controller modulates
the input to a system under control in order to steer the system and its output towards a desired reference value”. Essentially the control loop formed the backbone of the approach as it defines necessary components for resilience. They included a resilience target described by the resilience metrics, and defense measures are used to proactively maintain the network in the face of failure. Results of challenge analysis components, which detect and characterize failures in the network, are fed to a resilience estimator that determines whether the target has been met. Then from there the resilience manager controls mechanisms that can preserve, recover, or gracefully degrade parts of the network. Thus the resilience manager controls the recovery aspect of the approach (Figure 3.3). The remediation of the network is accomplished by the resilience manager that uses alerts from the detection component. The manager selects an adaptation strategy based on the information contained within the alerts. An adaptation strategy could be new path computations, topological configurations, channel allocations, or forwarding structures. Thus the mechanisms are deployed by pushing new configurations onto nodes of the network. The manager also assesses the success of its implemented strategies and uses the results to adapt [Smi11].

3.2.4 Building Secure, Resilient Architectures for Cyber Mission Assurance

Goldman et al. defines resilience according to the University of Kansas’s ResiliNets Project as, “the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation.” The focus is on achieving resilience against specific patterns of cyber attacks in mission-critical computing environments. The architectures described are designed to be a balanced combination of proactive mechanisms, detection capabilities, and adaptive responses.

A risk management approach is used to determine which capabilities of the network must be
resilient to what degree and against what threats. This allows for deciding proactive approaches using the existing mechanisms and procedures while also identifying gaps and determining trade-offs of alternative courses of action. The usage of risk analyses and dependency modeling helps to identify the best locations for additional safeguards or the best locations to defend from attacks. The risk management approach provides information for selecting the best response.

Virtualizing the infrastructure allows adaptive responses to be faster and more cost-effective. Virtualization allows for techniques such as replication, scaling, and isolation. It also allows a cost-effective approach to diversity, which then allows for higher degree of resilience. The agility of virtual machines allows for many of the approaches presented to be even more effective. A master image of how to configure servers can be virtualized to provide a quick way to recover systems that have been corrupted or are under attack. Using replication, reconstitution, or addition of additional virtual machines allows for resilience of the network during a denial of service attack. However in using virtualization, steps must be taken to ensure that new vulnerabilities are not created as a result of using virtual machines and a hypervisor.

The remaining resilience techniques can be used in combination to create a resilient network. Diversity, redundancy, randomness and unpredictability are all proactive mechanisms. Verifying the integrity of essential hardware and software functions of a virtual machine is essential in reconstituting the network successfully. For example, a known, good image can be pushed onto the virtual machine instead of replicating a running virtual machine or replacing it with a backup (this can also be used as a proactive measure). Goldman et al. refers to periodically refreshing to a known good image as “non-persistence”. A variety of adaptive responses can be used to start the recovery process. Using the information from the damage assessment capability, the system can choose the best course of action, like shutting down a part of the network to restart it instead of letting it degrade to a suboptimal operating state. The security policies can be made stricter during recovery to adapt to the attacks, or the compromised capability can be reconfigured to a more trusted minimized state with fewer nonessential functions.
3.3 Revised Resilience Architecture

Resilience has been defined in literature many ways, Haimes et al. summarize a few definitions for resilience as: “Resilience is the result of a system 1). preventing adverse consequences, 2). minimizing adverse consequences, and 3). recovering quickly from adverse consequences or the ability of the system to withstand a major disruption within acceptable degradation parameters and to recover within an acceptable time and composite costs and risks [Hai09].” However, among experts within the field of resilience, generally accepted definition is: the ability to provide and maintain an acceptable level of service when facing attacks and challenges to normal operation [Tri09] or the ability of a system to resist/minimize potential damage in order to maintain system state (fight-through) within an accepted operational level in response to any external threat or attack [Kha].

We believe a resilient network must be capable of not only predicting (anticipating) attacks, incorporating well-known technique (prevention and detection), but also take the necessary steps to endure an attack when compromised, while returning to normal (recover) operating capabilities. Then automatically update (adapt), so as not to succumb to the same vulnerabilities. This multi-layer approach of having multiple systems capable of working in conjunction as well as independently of one another will make the network infrastructure vastly more resilient to intelligent adversaries.

3.3.1 Anticipate

Ability to utilize data in such a way (from both traditional and non-traditional areas) to forecast the next attack.

3.3.2 Prevent

Measures taken to actively look for vulnerabilities within ones one system (e.g. penetration testing) to secure weaknesses before they are exploited (proactive use of resources to secure system). Intrusion prevention is an active and passive stage in security research where the overall goal is to actively take steps to prevent an attack. (e.g. hiring white hats to actively determine weaknesses in your organizations network or adding network flow encryption to prevent a man-in-the-middle attack).

3.3.3 Detect

Typically a reactive measure, systems designed to look for abnormalities, behavior, in network or signatures of known attacks. The detection phase of security research is predominately passive. The majority of detection techniques are designed to scan network traffic/packets in an effort to identify attack ‘signatures’ or suspicious (anomalous) activity. The detection phase is where Firewalls (pfSense, Cisco ASA, Juniper SRX), Intrusion Detection Systems (IDS) (e.g. Snort, Suricata, Bro, etc) and Intrusion Response System (IRS) are located. The IRS is a component of the IDS. The role of
the IRS is to choose the most appropriate means of responding to a particular threat, assuming the threat is detected.

### 3.3.4 Recover

System designed to endure and return network systems to a degree of functionality after an attack has occurred. Recovery is designed to function in a post attack environment with the intent of returning cyber resources to a robust state/restoring the system to a normal state of operation by compensating for a security violation, by eliminating or repairing its effects.

The process of restoring a secure state in a system after there has been an accidental failure or a successful attack. The process of restoring an information system's assets and operation following damage or destruction.

### 3.3.5 Adapt

Update system defenses so they will no longer be vulnerable to same type of attack or exploit. In some cases, Adapt, will have a second meaning. The ability to adapt in real-time to an attacker to continue to try new solutions in the event the initial 'response' is not successful.

By describing these next-generation cyber resilience architectures, we are able to clearly motivate that recovery is a key component in the development of any future resilient architectures. In the following section we provide the reader with the correct terminology in which to characterize security systems that incorporate recovery.

demonstrate the need for a separate recovery mechanism.

### 3.4 Recovery Terminology and Characteristics

Each recovery technique was thoroughly analyzed in an attempt to formulate a common vocabulary (terms and characteristics). These terms and characteristics are based on the design and operation of each technique. The following is a list of terms used for this taxonomy: Degree of Automation, Response Time, Response Selection, System Distribution, System Deployment, Attack adaptive, IDS Reliance, and Proposed Recovery Technique. (Note: In order for a technique to be considered network-related, it must have some interaction with the network. For example, pulling data from the network (e.g. packet captures, data streams) or associate/communicate with a device on the network (e.g. firewall, IDS, a separate node etc)).

#### 3.4.1 Degree of Automation

**Degree of Automation**: describes a type of action the system takes in response to a threat.
• Manual: the system waits for input from a human on how to recover from a threat.

• Semi-automated: the system is capable of low-levels of automated response, however the primary response/actions/decisions are made by system admins.

• Automated: the system is capable of responding autonomously without human intervention.

3.4.2 Response Time

Response Time: describes how the recovery method reacts to a threat.

• Delayed: the system takes no action until confirmation of the attack has been received; this confirmation is provided by the system administrators, sensors, or other confidence metrics on hosts or throughout the network. (Example: systems relying on checkpoints as fault tolerance mechanisms, delayed response might lead to inability of a system to roll back to a safe state.)

• Proactive: the system is capable of anticipating/playing out a result of an attack before the intruder has a chance to affect it. These systems are rare and difficult to implement as they rely on probabilistic measurements and analysis of user/system, social data, and other behavioral characteristics.

3.4.3 Distributed

Distributed: describes how the response systems interact with one another upon detection of a threat.

• Centralized: a singular system (host/node) that handles the recovery for the entire network (e.g. single host-based recovery system that is designated to a single machine).

• Distributed: a system that responds to an attack by combining the efforts of multiple ‘distributed’ systems throughout the network. In this instance, distributed means ‘more than one’ recovery component. While distributed systems are capable of functioning independently, their strength lies in their ability to coordinate with other systems before making a final global response. Unfortunately, distributed systems are more complex to implement and require stronger/robust communication with each component.

3.4.4 Level of Operation/Deployment

Level of Operation/Deployment: describes where the system is deployed.

• Network: deployment is located on traditional networking equipment or network ‘node’. Other variants of these systems use network resources to recover clients, servers, etc.
• Host: deployment is located on the end-user host/client. These types of recovery techniques typically rely on a host system to gather data regarding the state of the network to be used when ‘recovery’ is necessary.

• Both: deployment is located on both the network and the client. These systems are capable of recovering both network functionality and end users (global recovery).

3.4.5 Attack-Adaptive

**Attack-Adaptive**: a response and recovery system capable of trying different techniques in the event the initial response technique was not successful.

• Static: the system's response selection mechanism remains the same during the duration of the attack. To date, the vast majority of responses are static.

• Adaptive: the system is capable of dynamically adjusting the response selection to an attack in real time. The adaptation to the attack could be either the real time evaluation of the success/failure of previous responses or the auto adjustment/activation of additional IDS.

3.4.6 IDS Reliance

**IDS Reliance**: describes whether or not the recovery system requires an IDS in order to function or if the recovery system is capable of standing on its own.

• IDS Reliant: system is solely dependent on data/logs received by the IDS. This information is then used to trigger the recovery response. Additionally, these IDS logs dictate the steps and procedures used by the recovery device.

• IDS Independent: system capable functioning independently of the IDS. While there is some degree of interaction, it is not required to function (initiate recovery).

3.4.7 Response Selection

**Response Selection**: describes the general method used to address the threats.

• Static Mapping: A manual response system designed to map an alert to a predefined response [Sta07]. While static mapping easy to set up and implement, it lacks the ability to address the current state of the whole system and is predictable.

• Dynamic Mapping: A slightly more advanced mapping system that bases the intrusion response on numerous factors such as network policies, system state, and attack metrics (severity, confidence, frequency, etc). The ability to adjust these metrics makes dynamic mapping a
flexible alternative. However, the major drawback is that it does not learn, and therefore must be manually updated [Ali12].

- Cost-sensitive Mapping: A mapping system designed to address the issue of comparing intrusion damage against response cost. These factors are utilized in order to generate a cost sensitive model, which provides the optimal response. However, in order for this method to be effective, the cost factor values must be continuously updated, and in some cases manually [Sta07].

3.4.8 Recovery Technique

Recovery Technique: methods that are implemented to recover from an attack. While most of these methods are specific to their respective recovery technique, there are instances where multiple recovery techniques utilize the same method (note: This is only applicable to sole recovery techniques in Table ??).

- Distributed Hash Table: nodes of the network are stored in a distributed hash table based on specific attributes in order to prevent attackers from disrupting the entire network.

- Backup Topology: topology of the network or configurations of individual routers are stored in order to replace corrupted parts of the network as failures occur.

- State Based: functionality of the network is represented as states using graph metrics to measure continuity of operations (recovering the network as an optimization problem).

- Preferential Node Repair: Nodes are weighted based on their dependencies with other nodes and how critical they are to the functionality of the network, with the highest weighted nodes repaired first.

3.5 Review of Network Recovery Techniques

In order to organize and categorize the various recovery techniques, we provide the reader with two detailed tables: Table ?? and ?? Table ?? provides a list of Intrusion Response Techniques with a recovery component. Table ?? lists systems that are similar to those in Table 3.1, but are not reliant on an IDS.
Table 3.1 Current List of all intrusion response techniques that incorporate recovery

<table>
<thead>
<tr>
<th>IRS</th>
<th>Year</th>
<th>Selection</th>
<th>Time</th>
<th>Type</th>
<th>Adaptive</th>
<th>Distributed</th>
<th>OSI Level</th>
<th>IDS Reliant</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC&amp;A</td>
<td>1996</td>
<td>dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>CSM</td>
<td>1996</td>
<td>dynamic</td>
<td>both</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>EMERALD</td>
<td>1997</td>
<td>dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>BMSL-based</td>
<td>2000</td>
<td>static</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>SoSMART</td>
<td>2000</td>
<td>static</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>pH</td>
<td>2000</td>
<td>static</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
<td>host</td>
<td>yes</td>
</tr>
<tr>
<td>Lee's IRS</td>
<td>2000</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>AAIRS</td>
<td>2000</td>
<td>dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>adaptive</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>SARA</td>
<td>2001</td>
<td>static/dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>host</td>
<td>yes</td>
</tr>
<tr>
<td>CITRA</td>
<td>2001</td>
<td>static/dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>TBAIR</td>
<td>2001</td>
<td>static/dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>not defined</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Network IRS</td>
<td>2002</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Tanachaiwiwat's IRS</td>
<td>2002</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>semi-auto</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Spec.-based IRS</td>
<td>2003</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>ADEPTS</td>
<td>2005</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>automatic</td>
<td>adaptive</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>FLIPS</td>
<td>2005</td>
<td>static</td>
<td>proactive</td>
<td>semi-auto</td>
<td>static</td>
<td>autonomous</td>
<td>host</td>
<td>yes</td>
</tr>
<tr>
<td>FAIR</td>
<td>2006</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>automatic</td>
<td>static</td>
<td>autonomous</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Stakhanova's IRS</td>
<td>2007</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>semi-auto</td>
<td>adaptive</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>DIPS</td>
<td>2007</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Jahnke</td>
<td>2007</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Strasburg's IRS</td>
<td>2008</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>adaptive</td>
<td>autonomous</td>
<td>host</td>
<td>yes</td>
</tr>
<tr>
<td>FOREVER</td>
<td>2008</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>automatic</td>
<td>adaptive</td>
<td>distributed</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>RRE</td>
<td>2009</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>automatic</td>
<td>adaptive</td>
<td>distributed</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>IRDM-HTN</td>
<td>2010</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>semi-auto</td>
<td>static</td>
<td>distributed</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>OrBAC</td>
<td>2010</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>semi-auto</td>
<td>adaptive</td>
<td>distributed</td>
<td>network</td>
<td>yes</td>
</tr>
<tr>
<td>Kheir's IRS</td>
<td>2010</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>semi-auto</td>
<td>adaptive</td>
<td>distributed</td>
<td>network</td>
<td>yes</td>
</tr>
</tbody>
</table>
Table 3.2 Current List of other intrusion response techniques that are not reliant on the IDS

<table>
<thead>
<tr>
<th>System</th>
<th>Year</th>
<th>Method</th>
<th>Automation</th>
<th>Response</th>
<th>Distributed</th>
<th>Deployment</th>
<th>Adaptability</th>
<th>IDS Reliant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phoenix</td>
<td>2003</td>
<td>Distributed Hash Table</td>
<td>Semi-Auto</td>
<td>Reactive</td>
<td>Distributed</td>
<td>Hosts</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DIBS</td>
<td>2004</td>
<td>Distributed Hash Table</td>
<td>Semi-Auto</td>
<td>Reactive</td>
<td>Distributed</td>
<td>Hosts</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FPRCF</td>
<td>2007</td>
<td>Backup Topology</td>
<td>Manual</td>
<td>Active</td>
<td>Centralized</td>
<td>Network</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SRDN</td>
<td>2013</td>
<td>Time</td>
<td>Automated</td>
<td>Active</td>
<td>Distributed</td>
<td>Hosts</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ARCC</td>
<td>2013</td>
<td>State Based Hash Table</td>
<td>Automated</td>
<td>Active</td>
<td>Distributed</td>
<td>Network</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RINALA</td>
<td>2015</td>
<td>Preferential Node Repair</td>
<td>Manual</td>
<td>Reactive</td>
<td>Distributed</td>
<td>Network</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
3.5.1 Manual Recovery

The first recovery technique to be discussed is manual re-installation of the software. While not specifically listed in the table, this process is essentially genesis in terms of network recovery, and still remains one of the most widely employed methods by IT staff today. Manual recovery involves an IT staff member either remotely reimaging/purging or going to the physical location of the router, switch etc. to completely erase and reinstall the operating system through the use of back up or physical media (e.g. USB flash drives, CD/DVD, etc.) [Ger10]. An analogous example of such a recorded procedure, while not synonymous with the network infrastructure itself, would be M. Locasto’s records of an attack in March 2007, where the IT staff had to go out and perform manual installations on several servers and workstations in order to create a clean slate from which to clone onto other machines on the network. Manual installation/imaging is still the primary method used today, however it is typically done over the network, via network imaging, as opposed to physically going out to the equipment.

Ghosting/network imaging is another example of a manual recovery method. This process has become the most popular and widely used method to get networks up and running again once it is evident that it has been compromised. Ghosting/network imaging involves taking a copy from a healthy system and pushing the image out onto the compromised equipment. The image can either be a fresh network install, restoring the system back to an earlier state (assuming you have snapshots of the network), or merely pushing out patches and security updates onto the afflicted system [Gol11]. A good demonstration of such a technique, while not directly related to networking equipment, can be drawn from M. Locasto’s [Loc08] recorded examples of both the March 2007 and December 2007 attacks. In both of these cases, host machines were set up with new images and then the freshly wiped systems were used as masters to distribute their image across the network to the other hosts.

3.5.2 Damage Control and Assessment (DC&A)

DC&A, proposed by Fisch in 1996, is a dynamic mapping technique. DC&A is aimed at intrusion damage control and assessment whose primary role is the protection of local systems. This was truly one of the first dynamic mapping techniques specifically designed to respond to an intrusion. This technique uses a damage control processor and a damage assessment processor to dynamically select the appropriate response based on the suspicion level of the users activity that was provided by the IDS. This system was designed in such a way to dynamically adapt to the level of suspicion generated by the user and respond accordingly in real-time. For example, “during an attack DC&A would protect the computer from entering an irrecoverable state” [Fis97][MF00]. The system also was designed to conduct damage assessment after the intruder left the system or the systems had been cleaned in order to restore the system to the original state. In other words, DC&A ‘assessment’
procedure is able to ‘analyze’ log files and then replace stolen files from backup storage [Fis97] [MF00].

3.5.3 Cooperative Security Manager (CSM)

CSM is a dynamic mapping technique proposed by White et al. [Whi96] in 1996. In CSM, the selected response is based on the metrics associated with the attack and the information about the intruder’s behavior, which comes from the IDS/detection component. While CSM was not specifically designed to be proactive it can yield proactive reactions to intrusions in certain cases. For example, in a distributed scenario for individual hosts equipped with CSM combined together, to perform local intrusion, in addition to notifying other CSM equipped hosts about suspicious activity. As a result, the host can take proactive/preventative measures as opposed to waiting on a user. For example: “when attackers attempt to gain an unauthorized access to an account by trying different passwords, however instead of checking all possible passwords on one machine, attackers move to a different host after each failed attempt. It is a distributed IDS, CSM allows hosts to share information and detect intrusive user activity in a cooperative manner, however the response actions are determined and deployed by each machine local” [Whi96] [Sta07].

3.5.4 Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD)

EMERALD is a dynamic mapping technique proposed in 1997 by Porras and Neumann that determined the best response by evaluating the intrusion behavior and severity metrics from the IDS. EMERALD implements a distributed framework for automated response, intrusion detection, and network monitoring. Through the use of this layered framework approach, different abstract layers of the network can be monitored. When EMERALD responds to threats, it implements resolvers. It is the responsibility of the resolver to implement response policies. These resolvers analyze attack reports and coordinate response efforts. The overall behavior of these resolvers was similar to CSMs ‘distributed IDS’, where each resolver was responsible only for their specific level, however the resolvers would also share their information in a global response selection [PN97].

3.5.5 Lee’s IRS

Lee’s IRS: was the first cost-sensitive response technique to be proposed. It is designed to incorporate machine-learning techniques to automatically generate detection models that have been optimized for optimum resource efficiency. The cost-sensitive model looks at the operational cost (processing and evaluating data by IDS), damage cost (potential damage to a target should the IDS fail), and response cost (applying a response) to determine the best course of action when responding to an attack [Lee02] [Khe10].
3.5.6 Adaptive Agent-based Intrusion Response System (AAIRS)

AAIRS, while only presenting a theoretical foundation for a proposed automatic IRS, it was the most complex and dynamic mapping approach as of 2000. AAIRS employs a variety of ‘agents’ designed to fill a plethora of roles in the response process. However, computing the classifications and confidence level of new/ongoing attacks is reserved for the Master Analysis Agent. Once the Master Agent has finished, it then passes the task along to the analysis agent. The analysis agent then determines the best course of action. There are six potential actions available to AAIRS and they are classified by: timing, type of attack, type of attacker, degree of suspicion, attack implications, and the environmental constraints. Finally, AAIRS implements the tactics agent, in order to invoke the appropriate response. One unique aspect of AAIRS, aside from not requiring system administrator input after each event, is its ability to provide adaptation, which is done through confidence metrics and success metrics [Rag01].

3.5.7 Cooperative Intrusion Traceback and Response Architecture (CITRA)

CITRA presented in [Sch01] is another tracing mechanism combining both static and dynamic mapping techniques. The framework calls upon information gathered from security management systems, network IDS, and network infrastructure (firewalls, routers) to perform the following three actions: 1. Detect intrusion, 2. Trace intrusions back to their source, and 3. Coordinate local response actions based on the attack report. This response mechanism, like many of its predecessors, is based on degree of certainty and potential severity of the attack. Using these values, there is a predetermined set of responses/actions that may be taken. While the ‘response action’ is dynamically determined, CITRA still relies on a predefined set of responses. The architecture CITRA uses is based on a ‘neighborhood like structure’ to trace and respond to threats. It is within this architecture where the attack is submitted to a centralized authority. The ‘centralized authority’ or the Discovery Coordinator is in charge of determining the optimal system response and ultimately responsible for the overall global response. However, local agents (CITRA agents) are still capable of generating a local intrusion detection reports [Sch01].

3.5.8 Network IRS

Network IRS is a cost sensitive model, proposed by Toth and Kruegel that is a network-based response mechanism that builds dependency trees (first technique to implement this method) of the resources on the network in order to consider the costs and benefits of particular response actions. By utilizing a cost sensitive model, Network IRS is able to prioritize its responses. As a result, the Network IRS is able to maintain an extremely high state of operability. This is accomplished by the ‘response alternative’, which temporarily isolates a node. In this particular model a calculation is then performed on the isolated node in order to determine which responses have the least impact.
on the system and its services. In this model, the automatic response to an attack is to block the IP, with every service having a static cost \[ \text{Ali12} \]. When the network IDS detects an attack, the Network IRS algorithm looks to find the firewall/gateway that can be effectively used to minimize the penalty cost of the response. The proposed algorithm utilized by the Network IRS is designed to compare a situation where all resources are available to the ‘penalty cost’ of those resources being unavailable and the capability of that resource ability to function in a degraded state \[ \text{TK02} \].

### 3.5.9 Specification-based IRS

Specification-based IRS is a cost-sensitive model proposed by Toth and Kruegel \[ \text{TK02} \] and Balepin et al. \[ \text{Bal03} \] that performs the same cost comparison as Network IRS, but also attempts to model dependencies between services in the system. These models evaluate the impact of different response strategies on services and systems. The resources are handled in two different methods, on either a system map or resource hierarchy. In the model, Toth and Kruegel implement a dynamic method of adding new nodes for every type of alert generated by the IDS that were not previously seen on the system map \[ \text{Wan01} \]. These nodes represent a specific system object such as sockets, processes and files. Then, depending on the specific object type, a specific response action is taken; this enables a specific ‘cost’ to be assigned to each node \[ \text{Bal03} \].

### 3.5.10 Phoenix

Phoenix is a cooperative, distributed backup system that functions by selecting hosts at random to store backups of ‘cores’ as a model of distributed storage in a cooperative system. These ‘cores’ are hosts that share redundancy of attributes with other hosts, but are different from other cores to reduce the likelihood of redundant vulnerabilities. All the hosts and cores are stored in a distributed hash table according to the hosts’ attributes (operating systems, applications, and IP address). To preserve the privacy of users’ computers, each attribute is hashed and then stored in the table, allowing authorized users to unhash attributes and retrieve the necessary data. The only user interaction required is to allocate the appropriate amount of disk space for Phoenix. The recovery system is effective because it takes advantage of host diversity of a network. However, this means that a network with very similar hosts will not be as easy to recover as a diverse network. Unfortunately, the recovery time for a situation where many users attempt to recover files at once could be very high \[ \text{Jun03} \].

### 3.5.11 Distributed Backup for Local Area Networks (DIBS)

DIBS is similar to Phoenix in that it takes advantage of the unused disk space among connected machines to backup data. Each node manages its own list of files that contain critical files from itself and other nodes. DIBS reduced overhead comes from its ability to efficiently utilize unused disk
space in a local area network and to transfer data via peer-to-peer backups. DIBS also incorporates security and versioning functions ensuring that the user can retrieve an uncorrupt version of their file. The security feature helps prevent other users on the network seeing each others’ data in the case of an insider attack. DIBS has many other features that aid in efficiency such as synchronization, cleanup, reassignment, and more. The GUI allows for ease of use by users while ensuring all critical files are backed up [Hsu04].

3.5.12 Adaptive Intrusion Tolerant System (ADEPTS)

ADEPTS is a cost-sensitive, proactive, adaptive graph-based approach. The use of adaptive Intrusion Response using Attack Graphs was proposed by Foo et al [Foo05]. This method employs the use of intrusion graphs (I-Graph) to model attack goals in an attempt to determine the spread of the intrusion. ADEPTS relies on the IDS to map alerts/alarms to IGraph nodes. By creating these IGraph maps, ADEPTS is able to estimate the spread of the attack based on the confidence level of the alert. Modeling intrusions with attack graphs enables the identification of potential attack targets and as a result facilitates the user to make better responses. Using this approach, the selection process on how to respond to an attack is subsequently based on three factors: the effectiveness of a particular response to a previous attack, the level of disruption to normal uses caused by implementing the response, and like many preceding methods, the confidence level of the attack. ADEPTS, like AAIRS, is an adaptive response that utilizes an effectiveness index depicting how effective a particular response is against a specific attack [Foo05] [Ali12] [Sta07].

3.5.13 Distributed Intrusion Prevention System (DIPS)

DIPS is a cost-sensitive model implementing fuzzy logic for a real-time IPS proposed by Haslum et al. The model prediction module, in addition to its dynamic risk assessment are based on a fuzzy model. The fuzzy logic is used as a method of potentially substituting the ‘human’ variables ability to determine risk assessment in an attempt to capture and automate the risk estimation process based on a number of variables. Finally, the fuzzy automated inference system is designed to incorporate hidden Markov Models (HMM’s) to model interaction between network and intruder. Unfortunately, one of the major drawbacks of DIPS is that it is not capable of detecting DoS style attacks [Has07].

3.5.14 Jahnke

Jahnke et al. proposed a cost-sensitive, graph-based approach for modeling and comparing attacks against resources and the effect of the response taken as a result of the attack. Jahnke assigns values based on criticality and lethality. This approach is an extension of what was proposed in the Network IRS through the use of general, directed graphs focusing on different dependencies between resources and deriving quantitative differences between system states. By using this method, Jahnke
states that you are able to calculate the response positive effect difference by comparing two different graphs, one from before and one from after the reaction [Mar07].

### 3.5.15 Fast Proactive Recovery from Concurrent Failures (FPRCF)

FPRCF devises and evaluates a proactive recovery technique on local networks. It addresses the issues of more than one concurrent failure in connected networks by building off of multiple routing configuration (MRC), a technique for producing a set of backup network configurations in the event of a single failure. 2DMRC, the focused recovery approach, uses two levels of link weights: the original weights and the weight for restricted links. This attributes enable a simple algorithm (a heuristic algorithm that takes in a graph) for building backup topologies and a simple forwarding scheme [Han07].

### 3.5.16 The Fault/intrusiOn REmoVal through Evolution and Recovery (FOREVER)

The Fault/intrusiOn REmoVal through Evolution and Recovery (FOREVER) uses a two part approach to actually recover the network through the use of Replicas (Virtual Machines). These Replicas are cleaned (replaced) when either an attack is detected by the IDS or via timer-triggered periodic recoveries. FOREVER is also capable of updating itself in the aftermath of an attack (e.g. changing passwords and applying patches) [SO08].

### 3.5.17 Response Recovery Engine (RRE)

RRE is a cost sensitive, automatic recovery system proposed by Zonouz, Khurana, Sanders, and Yardley. RRE employs game-theoretical response strategies against advisories modeled as opponents in a two-player Stackelberg stochastic game. RRE applies attack-response trees to analyze undesired security events and their countermeasures using Boolean logic to combine lower level attack consequences, while accounting for uncertainties in IDS alert notifications. RRE then chooses the best response by solving a partially observable competitive Markov decision process. Initial testing observed that by scaling the ART trees, RRE could protect around 900 nodes on a network. However, at around 900 nodes, hardware limitations would begin to affect the overall calculations that could be made. This impairment of the speed of the calculations would directly affect the decision-making speed of choosing the most appropriate response towards an attack [Zon].

### 3.5.18 Spontaneous Recovery in Dynamical Networks (SRDN)

SRDN, while not necessarily a deliberate attempt to recover a network, allows for the affected network to spontaneously heal if given enough time. (Although this can be expedited by applying self-healing algorithms when designing a network). The SRDN aims to model 'spontaneous network
recovery’ by examining the ‘phase-flipping’ phenomena of local node recoveries. This is done by attempting to determine mathematical formulations for the average uptime and downtime of a system, which represents each phase of the system, the evolution of the system as a trajectory, and an average probability that any node has failed. By performing numerical simulations for regular networks, in addition to the mathematical formulations, it was determined that the system reduces to the Watts model if there are no recoveries. In the event dynamic recovery was introduced, the phase switching phenomena could occur [Maj14].

3.5.19 Autonomous Reconstitution of Compromised Cyber-systems (ARCC)

ARCC is an optimal and dynamic reconstitution approach. It utilizes a small set of cyber systems as graphs and a genetic algorithm for optimizing regeneration in addition to defining the systems as being in one of three states: fully operational, marginally operational, and compromised. In order to define the states as such, the state of a cyber system is assumed to be discrete. Resilience and continuity of operations metrics are used to determine what the next optimal state of operation should be achieved. Computing metrics in real time as the network is reconstituted is difficult, so indirect graph metrics such as algebraic connectivity and average path length are computed in real time and used as needed. The approach uses mathematical formulation as a step towards implementing an autonomous reconstitution algorithm. By treating the mathematical formulations as an optimization problem, a solution to the problem can be further developed [Ram13].

3.5.20 Recovery of Infrastructure Networks After Localized Attacks (RINALA)

RINALA uses a preferential node repair strategy by determining which compromised nodes have the highest priority and then attempting to repair nodes with the highest priority before moving on to other compromised nodes. RINALA demonstrates the deliberate recovery process of two-dimensional weighted square lattice using four different repair strategies and also applies their with local, random, and malicious attacks. Localized attacks represent natural hazards like earthquakes, cyclones, etc for a network like a power grid. Random attacks represent random failure of nodes or edges without deliberate interference. Malicious attacks represent deliberate attempts to damage the functionality of a network as much as possible. Several formulas are defined in order to describe the network functionality, recovery metrics, nodal weight distribution, and more. These are then used in their four recovery approaches: Preferential Recovery based on Nodal Weight (PRNW), Periphery Recovery (PR), Greedy Recovery (GR), and Random Recovery (RR). PRNW determines the node with the maximum weight among the nodes on a shortest path from the starting node, and the found node or edge is repaired first, then the process starts again. PR finds the nodes that are directly connected to the functioning part of the network and any damaged edges or nodes. The edge/node with the highest weight among the peripheral nodes is repaired, and then the set
Table 3.3 Intrusion Response Milestones

<table>
<thead>
<tr>
<th>IRS</th>
<th>Year</th>
<th>Selection</th>
<th>Time</th>
<th>Type</th>
<th>Adaptability</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC&amp;A</td>
<td>1996</td>
<td>dynamic</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>cooperative</td>
</tr>
<tr>
<td>CSM</td>
<td>1996</td>
<td>dynamic</td>
<td>both</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
</tr>
<tr>
<td>Lee's IRS</td>
<td>2000</td>
<td>cost-sensitive</td>
<td>delayed</td>
<td>manual</td>
<td>static</td>
<td>autonomous</td>
</tr>
<tr>
<td>ADEPTS</td>
<td>2005</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>automatic</td>
<td>adaptive</td>
<td>autonomous</td>
</tr>
<tr>
<td>RRE</td>
<td>2009</td>
<td>cost-sensitive</td>
<td>proactive</td>
<td>automatic</td>
<td>adaptive</td>
<td>distributed</td>
</tr>
</tbody>
</table>

of peripheral nodes and edges grows until it includes the entire network. GR determines the edge or node at any place that maximizes network functionality if repaired. While this method has the highest computational complexity, it is more effective in recovering high priority parts of a network. RR is the simplest method and choses a damaged edge or node randomly to be restored in each iteration [Hu16].

After conducting a comprehensive review of many of the key systems designed to recovery from a malicious attack from nearly two decades of literature, we can conclusively state that there has been very little research conducted on network recovery. Additionally, there has been even less research on recovering networks from malicious attacks. However, from the current research, we are able to glean insight into the shortcomings, progress, and research areas yet to be addressed in current methods. While the shortcomings will be address in Section 3.6, the achievements made by DC&A, CSM, Lee's IRS, AAIRS, ADEPTS, and RRE account for the major milestones to date. These six techniques were the first to propose and implement the key components in modern network recovery/intrusion response systems [DC&A, CSM, Lee's IRS, AAIRS, ADEPTS] as seen in Table ??.

3.6 Analysis of Recovery Techniques to Date

By conducting a thorough review of security systems that have incorporated a recovery technique in Section 3.5 we were able to conclude that there are virtually no systems in place designed to specifically automatically recover the network.

There is a tremendous over reliance of current recovery/response techniques on the IDS. There are two major shortcomings coupled with this over reliance on IDS: The first is that many of the proposed techniques assume a perfect detection rate, and of these detected/suspected attacks, that there is a 100% level of accuracy that every attack detected is really an attack [Foo05] [Us03]. The second is that the majority of Recovery techniques are considered to be a component of the IDS itself, which is a failing in and of itself. If your recovery/response technique is 100% reliant on system logs and data provided by the IDS in order to function, then what is the point of implementing
a recovery system? Scenario 1: In the event that the IDS detects an attack, the attack is blocked, keeping it from damaging the network. As a result, there is no need to recover. Scenario 2: When an attacker penetrates the IDS, the recovery technique is never activated, since the information on how to ‘recover’ comes from the attack logs and data provided by the IDS.

One the gravest issues that all of the proposed recovery methods overlook is the fact that they are deployed on the production network. As a result, these techniques assume that all of the routes are trustworthy. Trust, is the key component in all of the distributed recovery techniques. No recovery technique takes into account the damage a compromised router could do, especially during recovery. As a result, the minute a component on the production network becomes compromised; it renders all methods of recovery irrelevant.

The ability to automatically respond and adapt to attacks was, until recently, another weaknesses of the current recovery techniques. The ability to automatically recover from an attack either during or after is imperative. The shortcomings and potential damage that an intruder can cause while waiting for the system administer can be detrimental. It was not until ADEPTS, and to a greater degree RRE, that automation started to become common.

The final shortcoming of several of the previously mentioned recovery techniques comes from their hierarchical design structure. Designing a recovery/response technique based on a hierarchical structure results in a system design vulnerable to any single point of failure, rendering the entire system inoperable.

3.7 Discussion

With the traditional method of recovery described above, there were a variety of drawbacks such as: 1). The system is still reliant on human intervention 2). The recovery system is reliant on traditional means of detecting an attack. 3). This method is time consuming and unless there is sufficient redundancy within the network, will result in degraded network performance or even temporary network downtime while the ‘recovery method’ is deployed. The dimensions for succeeding here are to determine/classify: the basic service needs for operations and having knowledge of previous services that were operational prior to the attack. Then to determine what the best mechanism are for recovering using these dimensions.

3.8 Summary

One of the biggest problems in security research is how to recover the network during or in the wake of an attack. By conducting a comprehensive review of the literature we have concluded that the aforementioned security systems are not only ill designed, but also incapable of adequately providing a network recovery service, especially against modern persistent OSPFv2 attacks. In
Chapter 4, we evaluate the five primary recovery techniques (Restore, Rollforward, Reinitialize, Reconfigure, and Replace) that were gleaned from a thorough review of all well known security systems that incorporate recovery (which can all be found in the recovery milestones as seen in Table 3.3).
In this chapter, we examine the continued usefulness of the five recovery techniques (Restore, Roll-forward, Reinitialize, Reconfigure, and Replace) with the more modern persistent OSPFv2 attacks, by running actual attacks against an isolated network formed of typical commercial network elements, while allowing each such recovery mechanism to defend the network as seen in Section 4.4. Our experiments show that three of the five basic mechanisms can no longer defend against partitioning attacks when attacks are persistent, and provides relevant performance results for the other two. Our results also point the way to further improving these mechanisms for even more sophisticated attacks, in the future. The layout of this chapter is as follows: In Section 4.2 we first review general background and prior work and come up with the underlying common mechanisms that they utilize. Next, in Section 4.2.1 we describe the specific recovery studies we consider. Following this, in Section 4.3, we identify the five underlying elemental recovery mechanisms that are used in different combination by these recovery studies.
4.1 Introduction

The Internet represents a global infrastructure that is not only critical, but becomes more depended upon by various actors every day. As Metcalfe’s Law proclaims, the value of the network comes from providing wide connectivity, and we now depend on that connectivity to be global, and continuous. Implementing techniques to protect the network against malicious attacks has long been recognized as an important goal. There are many directions for such studies, e.g. recognizing and understanding vulnerabilities, protecting against specific exploitation of specific vulnerabilities, and detecting exploitation attempts. However, it is recognized that comparatively little attention has been paid in the literature to repairing the damage done by an attack while adapting to prevent further attacks [Zon14; Loc08].

While the Internet was architected and designed to provide redundancy and resilience to failure, routing attacks do not behave like ordinary failures. They intentionally cause routers to fail at a highly accelerated rate, or to propagate incorrect or inconsistent information throughout the network. Despite its robustness to normal failures, the operation of routing is vulnerable to deliberate attack (e.g. denial of service, eavesdropping, man-in-the-middle, and persistent OSPFv2 attacks) [Wu99; JLM04; Nak12]. The persistent OSPFv2 Partition attack is a recently published variance of these attacks [CN15]. Partitioning a network induces, in a sense, the worst failure of a network, since the connectivity that is the very hallmark of a network is lost. In this study, we consider recovery under this attack.

Our reasons for first extracting the five elemental mechanisms are as follows. First, different studies in the literature have used them in different combination, showing that these underlying mechanisms embody more foundational concepts that are worth understanding in depth, rather than specific heuristics that have been combined with them in different studies. Specific augmentations of these approaches may perform well for a particular network and attack scenario, but not in others; the fundamental mechanisms are more generally applicable. We hold that the insight obtained by studying the simpler, fundamental concepts, is the more valuable insight. We are also not in a position to precisely replicate such heuristics, or the exact manner of their combination, which are not precisely described by the authors in all cases.

Our contribution in this section is thus two-fold. We identify the fundamental mechanisms that have been used for network recovery against OSPFv2 attacks, and articulate them. We then use them in a real network, using commercial equipment, to determine and quantify their efficacy in the face of persistent OSPFv2 partition attacks. We have previously reported some of this work in [Mer18]. In this section, we provide additional experimental results, and describe the results more thoroughly; in particular, the study of the behavior of recover strategies under persistent DoS attacks described in Section 4.4.5 is completely new, and has not been previously reported in [Mer18] or elsewhere.
4.2 Background and Prior Work

The primary means of network security to date has relied on basic intrusion prevention and detection techniques. Intrusion prevention is both an active and passive stage in security research where the overall goal is to actively take steps to prevent an attack. The detection phase of security research is predominately passive. The majority of detection techniques are designed to scan network traffic/packets in an effort to identify attack signatures or suspicious (anomalous) activity. Detection systems include Firewalls, and Intrusion Detection Systems (IDS). An Intrusion Response System (IRS) is often a component of the IDS. The role of the IRS is to choose the most appropriate means of responding to a particular threat, assuming the threat is detected. That response may be to trigger a recovery mechanism, or simply to log data and/or notify administrators by raising an alarm. In contrast, the role of Recovery is to function in a post-attack environment with the intent of returning cyber (in this instance OSPFv2) resources to a robust state. Recovery is what is necessary when the attack is successful.

At present, the recovery of routers is primarily a manual process. Manual recovery involves an IT staff member reimaging a router to reinstall the operating system through the use of a system back up or physical media [Ger10]. Ghosting/network has become the most popular and widely used manual recovery method to get networks up and running again once it is evident that the network has been compromised. Ghosting/network imaging involves taking a copy from a healthy system and pushing the image out onto the compromised equipment. The image can either be a fresh network install, restoring the system back to an earlier state (assuming you have snapshots of the network), or merely pushing out patches and security updates onto the afflicted system [Gol11; Loc08].

4.2.1 Existing Recovery Studies

It is recognized in literature [Zon; Loc08] that while response systems have been addressed by a significant amount of research, there is a dearth of recovery systems in the literature, and recovery systems have not kept pace with modern attacks. Below we describe the significant works in this space.

Damage Control and Assessment (DC&A), proposed by Fisch in 1996 [Fis96], is a dynamic mapping technique. DC&A is aimed at intrusion damage control and assessment whose primary role is the protection of local systems. The system also was designed to conduct damage assessment after the intruder left the system or the systems had been cleaned in order to restore the system to the original state. In other words, the DC&A assessment procedure is able to analyze log files and then replace stolen files from backup storage.

Phoenix [Jun03] is a cooperative, distributed backup system that functions by selecting hosts
at random to store backups of ‘cores’ as a model of distributed storage in a cooperative system. These ‘cores’ are hosts that share redundancy of attributes with other hosts, but are different from other cores to reduce the likelihood of redundant vulnerabilities. All the hosts and cores (individual backups) are stored in a distributed hash table according to the hosts’ attributes (operating systems, applications, and IP address). These backups can then be accessed to restore a compromised node.

The Fault/intrusiOn REmoVal through Evolution and Recovery (FOREVER) [SO08] uses a two part approach to actually recover the network through the use of Replicas (Virtual Machines). These Replicas are cleaned (replaced) when either an attack is detected by the IDS or via timer-triggered periodic recoveries.

RRE (Response Recovery Engine) [Zon14] is a cost sensitive, automatic recovery system that employs game-theoretical response strategies. RRE then chooses the best response by solving a partially observable competitive Markov decision process. Initial testing observed that by scaling the attack-response trees, RRE could protect around 900 nodes on a network. However, at around 900 nodes, hardware limitations would begin to affect the overall calculations that could be made. This impairment of the speed of the calculations would directly affect the decision-making speed of choosing the most appropriate response towards an attack.

### 4.3 Fundamental Recovery Mechanisms

The literature in the previous section makes it clear that there are only several basic techniques employed by intrusion response systems to recover the network. Each technique was proposed in some specific research study, possibly with additional (less directly related to recovering the network) heuristic measures to increase the efficacy of the technique, while subsequent studies may have re-used or combined such techniques, or augmented them with different heuristics.

We identify the following five elemental mechanisms:

- **Restore:** Use incremental backups to return router to a previously known good state
- **Rollforward:** First Restore (as above) router, then find a new state from which the router can operate
- **Reinitialize:** Restart the router, OSPFv2 process, or links
- **Reconfigure:** Upload a different running configuration onto router
- **Replace:** Swap out compromised virtual router for a new virtual router with a different configuration

The literature described before may be seen to be using these elemental techniques as shown in Table 4.1. Below we provide more articulated discussion of each of these.
4.3.1 Restore

Restoring a router to its last known operational state is an assurance that as long as the issue is software related, the router will return to a functional state. The Restore technique involves either copying the most recently-known good running configuration into the router’s current running configuration or completely replacing the entire OS along with its configuration. Restoring a router to a previously functional state is still the most widely accepted and executed means of recovery [Jun03].

The Restore technique works by automatically making periodic backups of each router configuration on the network and storing them on a distributed file server. These distributed servers only store backups for the routers located in their designated zones (e.g. all of the virtual routers as seen in Figure 4.1 in Section 4.4 would use the same file server). Upon activation, Restore requests the last ‘configuration’ from the server, establishes a connection to the compromised router and replaces the compromised router’s current configuration with a backup configuration file. While ‘Restore’ remains the most widely employed technique, it is completely ineffective against malicious attacks against OSPFv2. This is for several reasons. The first of which is, in the case of a malicious attack, it is difficult to known when the last “known good state” would have been (i.e. before the router became compromised). Additionally, placing the same configuration back on the router will just result in the router being compromised again.

4.3.2 Rollforward

The Rollforward process is designed to recover the network by first restoring, and then using incremental log data to ‘Rollforward’ to get as close as possible to the most recent state before the router become compromised.

The Rollforward process uses both incremental and full backups to recover a router. These backups can be securely stored on a separate file server. When activated, Rollforward first requests the earliest full backup and then uses the incremental backups to make small updates right up until right before the router was compromised. Rolling forward could also involve updating the software such as the router’s operating system or drivers.

<table>
<thead>
<tr>
<th>Study</th>
<th>Restore</th>
<th>Rollforward</th>
<th>Reinitialize</th>
<th>Reconfigure</th>
<th>Replace</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Fis96]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[Jun03]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[SO08]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[Zon14]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4.1 Previous security systems utilizing these recovery techniques
The major downside of Rollforward (just as with Restore) is that the state right before a router was compromised is still vulnerable to the same persistent attack. This could cause an endless loop of trying to recover to the instance before attack.

4.3.3 Reinitialize

Reinitialize-Recovery is designed to refresh the individual link, the connections, or restart the OSPFv2 process the afflicted router. By refreshing the compromised routers links, the router will repopulate its OSPFv2 database. By reinitializing the OSPFv2 process on the afflicted router, the false routes and data from the the attack would be removed and the router would reform its adjacencies. An instance of this method can be seen in [Zon14].

Unfortunately, Reinitialize-Recovery is a temporary fix. Because of the persistent nature of the partition attack, once the router re-establishes its adjacencies, the corrupt LSAs will be propagated back onto the router.

4.3.4 Reconfigure

Attacks against the network, especially Persistent OSPFv2 attacks, rely on the static configurations and predictable behavior of routers. Therefore, by Reconfiguring a routers running-configuration (IP address, etc), the router will no longer be susceptible to that particular instance of the OSPFv2 attack that caused it to become compromised in the first place.

Reconfigure-Recovery works by automatically compiling potential interface addresses and storing them in a database. These ‘potential interface’ addresses are configured to be on the same subnet as the compromised router, with only nominal changes to their running-configuration such as the IP address. Reconfigure-Recovery pulls an IP address from the database corresponding to the compromised router, establishes a connection to the corrupt router, and configures each online interface to have a different IP address than before (Note: A different IP address must be assigned for each active interface on the compromised router). After a period of time, the OSPFv2 protocol will propagate the new interface address(es) to the rest of the network.

4.3.5 Replace

By Replacing a compromised virtual router with a virtual copy that has been preconfigured to be on the same subnet, OSPFv2 network, while having a different IP address etc, it is possible to restore network connectivity almost instantly without having the virtual-spare be susceptible to that same instance of the partition attack.

Replace-Recovery is specifically designed to work in a virtualized environment. It works by having a virtual-spare that is parallel to the production router with links that are shutdown. The configuration of the virtual-spare is nearly identical to that of the router it is meant to replace. It has
been configured to be in the same subnets but with different Router-ID and IP Addresses. In the event of an attack Replace-Recovery will automatically initialize the links on the router-clone and after waiting for the OSPFv2 database to populate, the compromised router is removed from the network. An example of this can be seen in [SO08].

One downside of this recovery technique is that administrators would be required to have a backup virtual router that is configured and ready to deploy in the event of an attack. It is difficult to predict where an attack may come from, but these backup routers can be strategically placed in locations on the network where there is only one route through the network. This method provides a form of recovery that minimizes network downtime and stops the persistent attack on the network.

4.3.6 Recovery Techniques and the Literature

It is thus apparent from the existing literature that any implementation of recovery can be seen as adaptations or enhancements of one or more of the five above approaches. Ghosting is specifically equivalent to Restore. DC&A is equivalent to a selective application of Restore, driven by a suspicion quantification algorithm. Phoenix applies Restore randomly to various routers. FOREVER implements visualization to deploy hot-swappable 'Reconfigured Replacements', while RRE applies Restore, Rollforward, Reconfigure, and Reinitialization.

4.4 Experimental Evaluation

This section investigates how effectively the five aforementioned techniques recover from a persistent OSPFv2 attack against the network. Subsection 4.4.1 first describes the lab conditions in which the experiments were run. Subsection 4.4.2 provides a brief overview of the attack chosen for the experiments. Subsection 4.4.3 shows the effectiveness of each recovery technique against the attack. Subsection 4.4.4 Evaluates each successful recovery technique (Reconfigure and Replace) in terms of the time it takes to recover, the overhead, and the quality of the data recovered. Lastly, Subsection 4.4.5 takes the successful recovery techniques (Reconfigure and Replace) one step further by evaluating them against a DoS attack.

4.4.1 Lab Setup

A development network was constructed using both physical and virtual routers. Both were commercially available production-class routers running the unchanged OSPFv2 protocol. The development network is illustrated in Figure 4.4. The network topology inside the cloud is chosen as it was the same topology used in Cohen et al. [CN15] to demonstrate the effectiveness of the Partition

---

1For convenience, virtual routers are labeled as VR# and are placed inside the cloud. The physical routers outside of the cloud are labeled as R#. 
Attack. The development network consisted of six physical (two Cisco 1921, two Cisco 3750s (with IP Routing enables), and two Juniper SRX 240s) and eleven virtual Cisco 7200 series routers running 15.2.4(M11). The virtual routers were hosted on a VMWare ESXi GNS3 [Gns] server.

Both of the major vendors were used in the construction of the testbed. This was for two specific reasons: The first of which was to demonstrate that these attacks are not vendor OS specific. The second reason was to emphasize that the attack exploits the LSA update process of OSPFv2, regardless of the vendors implementation (e.g. Cisco issues OSPFv2 LSA updates every default 30min per RFC-2328, while Juniper issues OSPFv2 LSA updates every default 50 minutes).

Each experiment was conducted under the same settings, with the only difference being the specific recovery technique under review. The three hosts connected to the network are virtual machines hosted on the ESXi server. Host 1 had the server deployment of network monitoring tools such as the Iperf\textsuperscript{2} [TU18] running on it. Host 2 acted as a client machine to gather data such as throughput and latency between Host 1 and itself. Host 3, the “Attack” VM, was used to launch attacks against the network. A custom built client/server tool, TCP-send was used to verify the connection between Host 1 and Host 2. The attack was then launched against vR1. Once the connection between the two hosts had been severed, a recovery technique was launched. A successful recovery was indicated by TCP-Send showing/re-establishing connectivity with Destination. Each scenario was run 15 times, to allow for variation of any underlying timing issues or other random factors. However, the results obtained were quite consistent, and below we present representative results.

4.4.2 Launching the Attack

The Partition Attack was launched from the designated ‘attack’ VM running an instance of Kali Linux. MD5 authentication was not implemented on OSPFv2 for two specific reasons. The first is that MD5 is vulnerable to key cracking, for which tools are readily available [Ren10]. Secondly, this attack is unaffected by LSA authentication, even if more powerful cipher protocol suites are used [CN15]. The attack works as follows to partition an OSPFv2 network into multiple parts. The attack starts with a single compromised router. This router (in this case vR1 as seen in figure 4.1) sends out a different LSA to each neighbor (R3 and vR2, and vR3), on different ports, with the same header information. The goal is to have each neighbor (R3, vR2, and vR3) create different routing maps of the network. The greatest strengths of the partition attack is its ability to overload remote links and routers and function even if the LSAs are digitally signed. The Partition Attack can result in forwarding loops, lengthened routing paths, and disconnected routers. Additionally, it is slightly more difficult to recover from because the routers no longer see a path to one another [CN15].

\textsuperscript{2}tool for measuring network performance
4.4.3 Effectiveness of each Recovery Technique

By evaluating the relative merits of these established recovery techniques, rather than the heuristics that may have accompanied them [Jun03; SO08; Zon14], only Reconfigure and Replace proved capable of recovering the network from the persistent OSPF Partition attack.

4.4.3.1 Restore

Restore was ineffective in recovering the network from the partition attack as seen in Figure 4.2. Even when 'Restore' was allowed to continuously run in response to each attack (run multiple times in succession), it still was not capable of recovering the network. In fact, it was observed that
Figure 4.2 Restore demonstrated no recovery

Figure 4.3 Rollforward demonstrated no recovery

this “looped-recovery” placed an increased load on the functioning portion of the network already saturated with looped packets destined for the unreachable address, compromising the network further.

4.4.3.2 Rollforward

Rollforward was also ineffective at recovering the network during the partition attack. Since *Restore* is the initial step in Rollforward, the results were identical to those of Restore as seen in Figure 4.3.

4.4.3.3 Reinitialize

Reinitializing the router interfaces and routing tables also proved ineffective in recovering the network, with the results also being identical to those of Restore. Additionally, it was observed (just
as with *Restore*) that Reinitialize, as seen in Figure 4.4, would actually degrade network performance further by adding increased traffic and churn to the network.

### 4.4.3.4 Reconfigure

Reconfigure was the first of the five techniques capable of successfully recovering the network. It was observed that the additional downtime following recovery was not the result of the attack, rather the result of the overall time it took for the routers to recalculate the new paths.

### 4.4.3.5 Replace

Replace was the most effective recovery technique. Unfortunately, this entire technique is reliant on a fully virtualized network and would not be a practical solution for traditional networks with physical
routers as it would require multiple physical routers at each location to act as the virtual-spare.

4.4.4 Quantitative Analysis for Reconfigure and Replace

This section provides a detailed quantitative analysis of the two successful recovery techniques, Reconfigure and Replace, using the following metrics: 1) The time it takes to successfully recover, 2) the performance impact of using Reconfigure and Replace, and 3) the quality of the data recovered.

The measurements were taken using a variety of client/server tools (with the client always being Host 1 and the server being Host 2). TCP-Send, was a custom built client/server tool used to send 64KB-TCP packets and measure the 'Syn-Ack' of the connection down to the microsecond. iPerf was used to measure the Bandwidth (BW) and the Latency (LAT). Due to the binary nature of this attack, it was determined that there was no-need to measure packet-loss (i.e. when the attack is successful, all packets are dropped, and when a recovery was successful, the transmission would resume). The data in the routing tables were compared (before, during, and after the attack) for inconsistencies. Finally, Traceroute was used to determine if the preferred path had changed as a result of the attack.

4.4.4.1 Time to Recover

The time to recover was the first metric used to evaluate Reconfigure and Replace, with the shortest downtime being the most desirable. TCP-Send was used to measure the overall time it took to recover the connect. This was calculated by taking the time at which the network first went down and subtracting it from the time the network resumed end-to-end connectivity. It was observed that on average, Reconfigure took 115 seconds of elapsed time to re-establish network connectivity as seen in Figure 4.5; while Replace consistently re-established network connectivity in 35 seconds as seen in Figure 4.6.
4.4.4.2 Performance Impact

The performance impact of Reconfigure and Replace were measured in terms of how much additional impact they had on the network in terms of bandwidth (BW) and latency (LAT). Prior to the first experiment, a base set of results for the network were established, indicating an average throughput of 943.4 Mb/s and latency of 31.87ms. These performance measurement were made without any attacks present on the network. The overhead for Reconfigure reduced the overall throughput by \(1.77\%\) or 16.75Mb/s from the base measurement and increase latency by \(1.57\%\) or .5ms, while Replace reduced the overall throughput by \(2.59\%\) or 24.45Mb/s from the base measurement and increased latency by \(0.94\%\) or .3ms as seen in Table 4.2.

4.4.4.3 Quality of Data

The Quality of Data within the routing tables after a recovery was an important metric to help gauge the success of the recovery. We wanted to determine whether or not there would be any traces of the false routes within the routing tables of the other routers post recovery, and if there were, if it would have a negative impact on the network. This information was gathered by comparing the routing table entries in the connecting routers before, during, and after the attack.

After Reconfigure reestablished network connectivity, it was verified that no false routes remained in the OSPFv2 routing database. This was not the case for Replace. It was observed that even after the network connection had been reestablished, the router that was Replaced (in this case vR1) still maintained the incorrect routes in its routing table. These routes remaining in the routing tables did not come as a surprise, as all routes will remain in the OSPFv2 routing table until they time out (expire) per RFC-2328 or are manually purged.

<table>
<thead>
<tr>
<th></th>
<th>Throughput (Mb/s)</th>
<th>Latency (ms)</th>
<th>Recovery Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>943.35</td>
<td>31.9</td>
<td></td>
</tr>
<tr>
<td>Reconfigure</td>
<td>926.6</td>
<td>31.4</td>
<td>115</td>
</tr>
<tr>
<td>Replace</td>
<td>918.9</td>
<td>32.2</td>
<td>35</td>
</tr>
</tbody>
</table>

4.4.4.4 Comparative Topology Analysis

To verify that the behavior observed is not in any way peculiar to the particular topology we happen to be using, we re-run the experiments on a second topology as shown in Figure 4.7. Since in our earlier experiment we already validated our tests on real-world hardware and virtual routers, for this
latter topology we simulate the network using GNS3; The attack and recovery methods run similarly.

![Figure 4.7 USNET 24Node Topology](image)

From our simulation, we are able to demonstrate that the methodology for implementing these recovery techniques holds for multiple topologies.

As can be seen in Figure 4.8 and 4.9, there is a slight increase (7s) in the amount of time either
4.4.5 Quantitative Analysis for Reconfigure and Replace against a DoS Attack

This section provides a detailed quantitative analysis of the two successful recovery techniques, Reconfigure and Replace, from a DoS attack against an individual router. Both Reconfigure and Replace were tested using the same protocols outlined in Section 4.4.1. This included the multi-vendor deployment of the Development Network with both physical and virtual routers and the GNS3 router simulation of US-Net. Each scenario was run 15 times, with the lab set up based on the topology in Figure 4.1 and US-NET in topology Figure 4.7.

A DoS attack was chosen because the intent of a DoS attack is not network destruction, but rather network disruption through performance degradation. However, one similarity between the two is that both of these attacks are persistent in nature. As such, it is recovering from the 'persistent' nature of the attack that is the focus of this second study, rather than the type of attack itself. The particular DoS attack that was chosen was a Python based DoS attack. The attack was specifically designed to spam an OSPFv2 router with 'hello' messages, eventually exhausting the routers resources. It was observed that under duress, from a single persistent attacking source of this particular DoS attack, will decrease network throughput by roughly 25% with an average latency of 51.47ms (by comparison to our initial base measurements) as seen in table 4.3).

Finally, just as in Section 4.4.4, the following metrics were used: 1) The time it takes to successfully recover, 2) the quality of the data recovered, and 3) a Comparative Topology Analysis.
Table 4.3 Recovery Overhead and Time During DoS Attack

<table>
<thead>
<tr>
<th></th>
<th>Throughput (Mb/s)</th>
<th>Latency (ms)</th>
<th>Recovery Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>943.35</td>
<td>31.9</td>
<td></td>
</tr>
<tr>
<td>DoS Attack</td>
<td>707.51</td>
<td>51.47</td>
<td></td>
</tr>
<tr>
<td>Reconfigure</td>
<td>694.99</td>
<td>52.28</td>
<td>115</td>
</tr>
<tr>
<td>Replace</td>
<td>689.19</td>
<td>51.98</td>
<td>35</td>
</tr>
</tbody>
</table>

4.4.5.1 Time to Recover: DoS Attack

Once again, the time to recover was the first metric used to evaluate Reconfigure and Replace. TCP-Send was used to measure the overall time it took to re-establish network connectivity. This was calculated by taking the time at which the network first went down and subtracting it from the time it took for the network to resume end-to-end connectivity. For both Reconfigure and Replace, it can be observed at the first mark where the DoS attack was launched and began to affect the network through a reduction in overall performance. However, once recovery was activated, all network connectivity was temporarily lost until Recovery was complete. Once again, it was observed that on average, Reconfigure took on average 115 seconds of elapsed time to re-establish network connectivity as seen in Figure 4.10; while Replace consistently re-established network connectivity in 35 seconds as seen in Figure 4.11. By comparison to our initial run against the Partitioning attack, it was observed that the recovery times were indeed identical.
4.4.5.2 Quality of Data: DoS Attack

Just as in Section 4.4.4.3, the Quality of Data within the routing after a recovery was an important metric to review. In the instance of Reconfigure, it was observed that the remains of the address hosts that were launching the attack still remained in the router. However, since the connection how now been terminated with the Reconfiguring of the routing address, these hosts will eventually time out (expire) as per RFC-2328. This was not the case with Replace, since replace was an entirely new virtual router, it did not have any stored information from the attacking hosts.

4.4.5.3 Comparative Topology Analysis: DoS Attack

Once again, to verify that the behavior was not specific to the topology we were initially using, we re-run the experiments on a second topology as shown in Figure 4.7. We again use GNS3 to simulate the network on which we launch the attack.

Once again, from our simulation, we are able to demonstrate that the methodology for implementing these recovery techniques holds for multiple topologies.

As can be seen in Figure 4.12 and 4.13, there is a slight increase (7s) in the amount of time either Reconfigure or Replace takes to reestablish network connectivity, but this can probably be simply attributed to the larger diameter of the network. The nature of the main results were the same for both topologies.

4.5 Discussion and Limitations

By implementing and studying the effect of the five recovery techniques, we have determined that:
• Restore, Rollforward, and Reinitialize are not capable of recovering an OSPFv2 network from a modern OSPFv2 attack

• Reconfiguring a routers running-configuration (changing the IP address, etc) allows the router/network to recover. This is because the configuration on the targeted router is no longer in sync with that of the attacker.

• Replacing a compromised virtual router with a virtual-spare that has been preconfigured to be on the same subnet, OSPFv2 area, while having a different IP address, etc allows the router/network to recover. This is achieved by having the configuration of the virtual-spare no longer in sync with that of the attack.

In hindsight, the inability of some of these techniques to address persistent attacks is not as
surprising. The majority of the recovery techniques (Restore, Rollforward, Reinitialize, Reconfigure) were never designed to recover from an intelligent adversary with persistent malicious intent.

Reconfigure and Replace were the only two established recovery techniques capable of restoring network connectivity. Each of these technique were able to recover from both the persistent nature of the attack, and the exploitation of OSPFv2’s LSA update.

The scalability aspect of Reconfigure could be a double-edge sword. Because Reconfigure only requires storing a slightly altered text configuration file for each router on the network, it is relatively easy to create such a list. The major limitation of Reconfigure would be with larger networks. The larger the network, the longer it would take for the changes to propagate throughout the network. Additionally, for a scenario where routing is enabled on a 24/48 port switch, significant network architectural planning would have to be made in advance to assure spare-link availability, possibly through the implementation of multiple VLANs. Reconfigure thus comes with a certain raised network planning cost.

Replace was by far the most successful of the established recovery techniques. It restored network connectivity 80 seconds or 228% faster than Reconfigure and was the only technique specifically designed to address malicious threats to the network. Conceptually, Replace is the same as Reconfigure; thus its strength comes from incorporating a virtualized environment, where a virtual router may be hot-swapped for another. Replace may not be a practical backwards compatible solution for traditional networks without any virtualization. Virtualizing the routers makes Replace an extremely scalable recovery solution, but also costly.

Finally, as always, the adversary does not stand still. As networks adopt Replace or similar approaches for robust routing recovery, it is likely that persistent attacks will also grow more sophisticated. It is easy to imagine a scenario in which the persistent attacker monitors OSPFv2 traffic, and attempts to follow the virtual router swaps. This further underscores the fact that more work is needed on recovery. The goals of such work must be reduce overhead and improve the flexibility of routing recovery, while attempting to stay one step ahead of an increasingly sophisticated persistent attacker.

4.6 Summary

In this chapter, we have presented five elemental recovery techniques that are seen to underpin routing recovery approaches in the literature, and evaluated their effectiveness in recovering from a modern attack against the routing infrastructure. Only some of the recovery techniques proved successful in re-establishing network connectivity.

In the following chapter, we introduce a new technique known as the Virtual Router Resiliency Cluster (VRRC) to make OSPFv2 more resilient to persistent attacks. We demonstrate that by combining the right set of components it is possible to successfully detect (and recover from) all known
persistent OSPFv2 attacks.
Routing protocols, including OSPFv2, are essential for correct network function and therefore attractive targets for attackers. Many attacks on OSPFv2 are known. Limited progress on detection and prevention has meant that protecting against and recovering from such attacks typically requires significant manual effort. The goal of this research is to make networks more resilient to all known attacks that exploit OSPFv2 Link State Advertisements (LSAs), without requiring modifications to the OSPFv2 protocol.

The proposed solution adds a Virtual Routing Resilience Cluster (VRRC) to the network infrastructure. This cluster makes use of three concepts. The first is that new links are added infrequently to networks. By comparing with previous topology information, suspect LSA information can be detected and rejected. The second is that it should be possible to corroborate LSAs by comparing with data made available from other layers of the network. LSAs that conflict with data from other layers can safely be rejected. The third is a rudimentary recovery and adaptation system designed to reestablish network connectivity and prevent the attack from happening again. This can be accomplished by purging the routing table and generating routing policies to block the offending source of the attack.

A VRRC was added to a testbed consisting of standard routers, and shown to provide protection against state of the art attacks. The impact of the implementation on network throughput and router
This chapter is organized as follows. Section 5.2 summarizes the operation of OSPFv2, reviews the known (persistent) attacks against OSPFv2, and summarizes prior work on defenses against such attacks. Section 5.3 proposes a new approach to surviving attacks that involve false link state advertisements. Section 5.4 describes an implementation of this approach, and an evaluation of its effectiveness and costs. Section 5.5 discusses the benefits and limitations of the proposed approach, and section 5.6 summarizes the chapter.

5.1 Introduction

Increasing network resilience to malicious attacks has long been recognized as an important goal, but most network security research has focused on other areas, e.g., recognizing and understanding vulnerabilities, protecting against exploitation of vulnerabilities, and detecting exploitation attempts. Much less attention has been given to repairing the damage done by an attack while adapting to prevent further attacks in order to continue operations [Zon14; Loc08].

Malicious attacks intentionally cause routers to fail at a highly accelerated rate, or to propagate incorrect or inconsistent information throughout the network [Gol11]. Despite how robust the distributed operation of routing is to normal failures, routing is vulnerable to deliberate attack (e.g., denial of service, eavesdropping, and man-in-the-middle) [Nak12; Jon; Jou00].

It is desirable for any method that improves network resiliency to do so without requiring modification to existing protocols (which requires a lengthy period of consideration / adoption / propagation), to preserve the ability to dynamically and automatically upgrade the network, and to impose very modest cost and performance overhead.

In this section we propose to add a new network component called a Virtual Router Resilience Cluster (VRRC). The goal of the VRRC is to detect, recover, and adapt to all known attacks involving OSPFv2 Link State Advertisements (LSAs). VRRC makes use of 3 concepts to improve the resilience of the OSPFv2 protocol:

- Comparison of LSA information with previous knowledge of network topology and state. This narrows down the nature and source of the attack [Wu99].

- Comparison of LSA information with data gathered from other, lower level protocols, such as Link Layer Discovery Protocol (LLDP) [Mon07].

- Basic Recovery [Mer18] and Adaptation by purging routing tables to eliminate false routing information and the generation of automatic routing policies to block the offending source.

By implementing these techniques, it was determined that:

- Analyzing Link-State IDs, Router IDs, and Advertising Router IDs allows detection of attacks.
• Analyzing data gathered from other layers, unaffected by OSPFv2 attacks, can be used in attack detection.

• Purging routing tables has benefits. Corrupted routes have to be reinstalled, with increasing likelihood of attack failure or detection / identification. In addition, the generation of automatic routing policies will prevent the same source from continually being used to launch an attack.

Our contribution is thus twofold: to articulate and demonstrate the effectiveness of the three techniques we listed above, as well as demonstrate an operational network component VRRC, which combines these techniques.

5.2 Background

OSPFv2 has several components that make it very robust against network failures such as flooding, the fight-back mechanism, and link authentication [Wu99; Moy98a]. Unfortunately, these techniques do not necessarily protect against persistent OSPFv2 attacks such as the Remote False Adjacency, the Disguised LSA v.1 (D-LSA v.1), Disguised LSA v.2 (D-LSA v.2), and the Partition Attack, which are the focus of this section. [Jon; Nak12; Nak14; CN15].

5.2.1 Remote False Adjacency

With the Remote False Adjacency (RFA) attack (also known as the Periodic Injection Attack presented by [Jon]), the attacker impersonates a phantom router located on the local network of a victim router (R3 in the figure). The victim router then advertises on behalf of the local network, an LSA containing a link to the phantom router. The consequence of this attack is that the attacker can confuse R3 into establishing adjacencies to the phantom router on the victim's local network, essentially black-holing the router [Nak12].

5.2.2 Disguised LSA v.1

The Disguised LSA v.1 (D-LSA v.1) exploits a weakness in the way OSPFv2 calculates the LSA. In OSPFv2, LSAs are “considered Identical if they have the same Seq. #, Checksum, and Age” [Moy98a]. The D-LSA v.1 attack exploits this calculation by advertising a disguised LSA that matches a recently generated (legitimate) LSA, that has yet to be installed by all routers in the AS. The attack works as follows:

1. An attacker sends a spoofed LSA of R3, to R3 (this is known as the “trigger”)

2. Simultaneously, the attacker sends a disguised LSA of R3 to R5, mimic’ing a “fight back” LSA from R3
This will result in R3 sending a “fight back” to R5. However, R5 will reject this legitimate update, instead using the more recent disguised LSA that was placed in its routing table from step 2. Finally, R5 re-floods the disguised LSA to R3, causing R3 to update its routing table with the now falsified data from R5; the result of which is route poisoning [Nak12].

5.2.3 Disguised LSA v.2

The D-LSA v.2 is a variant of the D-LSA v.1 attack. When a router calculates what links should reside in its database, it only uses the Link State ID field, as opposed to all three fields (LS Type, Advertising Router, and LS-ID). The attack works as follows:

1. An attacker sends a spoofed LSA of R3, to R3 (“trigger” a fight-back) to learn: Link State ID, Advertising Router, and Sequence #.

2. The attackers sends a LSA to R5, where:
   
   \[
   \text{Link State ID} = \text{ID of R5} \\
   \text{Advertising Router} \neq \text{ID of R5} \\
   \text{Sequence # of R3} > \text{sequence # of R5}
   \]

   Nakibly et al. were the first to develop an exploit such as this, based on the ambiguity in the process OSPFv2 uses for the formation of its LSA databases. It was concluded that the way a router
chooses the appropriate LSA to install into its database must be “implementation/vendor” dependent. The consequences of this attack are: Link overload, Long Routes, Delivery failure, Routing Loops, Churn, valid LSAs replaced with false LSAs, routing tables of all routers poisoned, and the routing table of the victim being erased [Nak14]. While D-LSA v.2 has been patched (table 5.1), it was still included as a metric for testing due to its ability to mimic new (legitimate) links being added to the network.

### 5.2.4 Partition Attack

The Partition Attack is the most recent attack against OSPFv2, designed to partition the network into two parts. This is accomplished by having a compromised router (in this case R3) send out a different LSA to each neighbor (R2 and R5), on different ports, with the same header information. The goal is to have each neighbor (R2 and R5) create different routing maps of the network.

One of the greatest strengths of this attack is that it works even if the LSAs are digitally signed. The author states that, “Judicious use of ‘self-LSA’ falsification attacks can overload remote links and routers that are far from the attacker, while leaving no obvious data plane traces that may lead to the attacker.” The Partition Attack can result in forwarding loops, lengthened routing paths, and disconnected routers. Additionally, it is slightly more difficult to recover from because the routers no longer see a path to one another [CN15].

Table 5.1 is included to provide the reader with a summary of the aforementioned attacks. Prior work on defenses against OSPFv2 attacks include [WW98; Wu99; Nak12; Hon15]. For convenience, the categories are:

- **Persistent**: is the attack natively persistent?
- **Patched**: has the attack been patched by the vendor?\(^1\)
- **Signed**: would enabling OSPFv2 MD5 authentication, and configuring keys correctly, defeat such an attack from untrusted routers
- **Proposed Technique**: reference to previously-proposed solutions

Several studies have been conducted on resilient networks, e.g. [Smi11]. Most of this prior work does not address malicious threats. Additionally, most solutions involve either making significant changes to the network infrastructure or to the protocols themselves. In the case of attacks against OSPFv2, specifically exploits against LSA, several solutions were proposed by [WW98] suggests the use of a secure wrapper with different levels of checking gates for input and output between the secure wrapper and the protected router. The secure wrapper examines all data, using a security policy database to determine the status of these checking gates. [Wu99] introduces a scalable IDS

---

\(^1\)Verified in lab through testing against recent copies of Junos and IOS

67
Table 5.1 OSPFv2 Attacks

<table>
<thead>
<tr>
<th>Attack</th>
<th>Persistent</th>
<th>Patched</th>
<th>Signed</th>
<th>Prop Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rem False Adj</td>
<td>Yes</td>
<td>Yes-[JSA16]</td>
<td>Yes</td>
<td>[Nak12]</td>
</tr>
<tr>
<td>D-LSA v.1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[Nak12]</td>
</tr>
<tr>
<td>D-LSA v.2</td>
<td>Yes</td>
<td>Yes-[CSA13] [JSA13]</td>
<td>Yes</td>
<td>[Nak14]</td>
</tr>
<tr>
<td>Partition Attack</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>[CN15]</td>
</tr>
</tbody>
</table>

incorporating a ‘detection’ module that performs statistical and protocol-based intrusion analysis and allows the user to remotely modify the ‘prevention’ and ‘detection’ modules based on the detection information. [Nak12] discusses two potential measures that could be taken to prevent the remote false adjacency and disguised LSA. The remote false adjacency could be thwarted if the AS ensures that different links use independent secret keys. This solution requires complex and error prone key management practices. In the case of the D-LSA v.1 attack, it was proposed that one could mitigate the predictability of the LSAs by advertising false links with random values to randomize the LSA checksum. Unfortunately, this technique would also be impractical for deployment in real-world applications because it would ultimately result in increased routing table sizes.

5.3 Methods

The goal of the proposed system is to collect information about routers that can then be used to verify the link state advertisements. This approach uses data gathered directly from routing tables, and data from other protocols. Data gathered solely from routing tables is sufficient to combat some but not all attacks (e.g., D-LSA v.2). Based on this information, a recovery mechanism is proposed that ensures that techniques compatible with OSPFv2 will purge false information and update routing policies to block the falsified address.

The criteria for a satisfactory solution includes the following:

- Requires no changes to the OSPFv2 protocol
- Supports dynamic, non-malicious changes to the network
- Compatible with existing network infrastructure
- Reasonable overhead and cost

5.3.1 The Virtual Routing Resilience Cluster (VRRC)

The VRRC can be configured for both a centralized and decentralized deployment. A centralized deployment would be recommended for smaller networks while a decentralized would be better
suited large networks. This section focuses on the centralized variant of the VRRC. The VRRC consists of three components: Detect, Recover, and Adapt all configured to share the same physical interfaces. For this deployment of the VRRC, we assume a non-hierarchical, single area network. However, we do not make any assumptions about the topology of the network for deployment. The detection component creates a database from two sources of information: layer 3 OSPFv2 LSA updates (LSA Inspection) and layer 2 LLDP information (Multi-Layer Authentication).

LSA Inspection operates directly on the virtual router and is responsible for identifying and comparing the: Link State ID, the Router ID, and the Advertising Router ID. Concurrently, the layer 2 LLDP information for Multi-Layer Authentication is gathered on a virtual SNMP server. The virtual SNMP server resides on a dedicated VLAN allowing each physical router to have a direct connection to the SNMP server. The virtual SNMP server is configured to gather all of the LLDP information for all routing devices and any directly connected neighbor the routing devices may have. The data gathered from LLDP on the SNMP server is then compared against the LSA updates to verify that the LSA updates are indeed legitimate.

The Recovery and Adapt components of the VRRC utilize three virtual routers. These virtual routers are designed to replace the core routers on the network backbone as all traffic in the network would be routed through these devices. For the particular subnet in which the VRRC resides, the virtual routers are configured to always be the designated router and the backup designated router. The VRRC and any directly connected adjacent interfaces must constitute a single subnet. Finally, the virtual routers are capable of performing real-time consistency checks upon each other by utilizing information found within their link state databases (e.g. Link State ID, Source Router ID, and Advertising Router ID).

The Recovery component of the VRRC activates once a threat has been detected. The VRRC will then purge the network by forcing an OSPFv2 LSA update to have the network reestablish their routes. Once the LSA updates have been issued, the VRRC may not force an LSA update again until: 1) a different LSA update with a different set of falsified addresses is detected or 2) human intervention acknowledging the threat has been resolved. This will help prevent an attacker from exploiting the recovery component of the VRRC by potentially forcing the VRRC into an endless state of recovery.

The Adapt component of the VRRC has the ability to prevent the network from being susceptible to the same attack from the same source while simultaneously alerting system administrators of the apparent threat. As soon as an exploit has been detected, the VRRC will generate routing policies through the use of prewritten scripts. These scripts are used to update the routing policies on both the virtual and physical routers. These policies are structured to do two things, temporarily block OSPFv2 link state updates on the specific interface that originates the falsified LSA while simultaneously alerting system administrators of the apparent threat. The routers receiving the corrupt information would also receive a similar policy, stating "ignore a LSA that has the following originating router ID information contained in it." By issuing a policy that only temporarily blocks
link state updates from the originating interface (e.g. on a spoofed router), traffic is still capable of traversing the network, as the actual transmission of data between end users is not reliant on link state updates for actual transmission of data. Therefore the link continues to transmit traffic, yet the persistent nature of the attack is eliminated while system administrators investigate the attack vector. This also prevents the adapt component of the VRRC from being exploited to potentially block the transmission of legitimate traffic.

### 5.3.2 Information used for Detection

Once initialized, the VRRC enters "learning mode," a process which is only required once. The length of this phase is proportional to the size and complexity of the network. LSA Inspection forces several ‘0-timer’ back-to-back LSA updates. Routers use these LSA updates to begin to populate and build the normal OSPFv2 link state database. Concurrently, the SNMP server constructs its LLDP database consisting of layer 2 information from all routing devices in the network. The information provided by layer 2 allows the VRRC to be deployed on a production network without having to make any assumptions about the current state of the network.

Following the learning phase, the network state is prepared for attacks and production use. The information used to detect the presence of persistent OSPFv2 attacks are the Link State ID, the Router ID, the Advertising Router ID, and the MAC address from layer 2. The Layer 3 information is stored in an LSA database on the VRRC physical host, while the layer 2 data is stored on the VRRCs virtual SNMP server.

For layer 3, the following information is used:

![VRRC Architecture Diagram](image)

**Figure 5.2** VRRC Architecture
• The LSID field, which provides a unique self-identifier for an LSA [Moy98b].

• The Source Router ID, which uniquely identifies the router within the OSPFv2 routing domain. The OSPFv2 Source Router ID, while not required, is typically assigned to be one of the router's IP addresses.

• The Advertising Router ID, which identifies the originating router of an adjacency [Moy98b].

For layer 2, the Link Layer Discovery Protocol (LLDP) [Mon07] is used to gather data that can be used to cross reference against the Layer 3 data. LLDP is a universal protocol that is not vendor specific. LLDP allows lower level devices to learn about the capabilities and characteristics of LAN devices such as IP addresses, system name, MAC addresses, etc.

5.3.3 Detection

The VRRC captures traffic on each physical interface and by incorporating a unique set of filters, maps the three fields listed above to Boolean variables that check for an attack. These variables help to populate the database and do not permit adding an entry for an already existing entry. Once the device transitions into attack-mode, the VRRC is configured to always assume that attacks are legitimate, however if all of the data within the LSA update and the Layer 2 mapping matches the criteria of the algorithm, then the attack is set to ‘false’, and the LSA is verified as legitimate. LSAs as they are received are examined and compared with the information that was learned. The comparisons are described in detail in the following sections.

5.3.3.1 Real-Time LSA Inspection

OSPFv2 routing relies on LSAs to maintain and update the routing databases throughout the AS. If it is possible to provide an appropriate set of checks for LSAs, then it is possible to verify the legitimacy of these LSA updates, assuming a static network topology. In this instance LSA updates are verified by analyzing the Link State ID, the Router ID, and the Advertising Router ID.

The Link State ID (LSID) is the first of piece of information used by LSA Inspection. By leveraging the data within the LSID field, it is possible to detect most attacks that are capable of circumventing the fight-back mechanism. In addition, it allows detection of phantom routers (which can be the result of such attacks) by looking for erroneous LSIDs. The second piece of information used is the Router ID. The inspection of the Router ID is specifically designed to look for the D-LSA v.1 attack because the D-LSA v.1 uses a different Router ID. The final piece of information used by LSA Inspection is the Advertising Router ID. This information helps with the D-LSA v.2 attack, which has the same LSID and Router ID as other LSAs, but has a different source / Advertising Router ID. LSA Inspection provides resilience to any static OSPFv2 network infrastructure and can successfully
counter all known threats against OSPFv2 when the network is not changing and deployed in a purged OSPFv2 environment. However, when deployed on an active network without purging the databases will detect conflicting LSAs when attacks such as D-LSA v.1 occur. Unfortunately, it is impossible to differentiate between the legitimate LSAs and falsified LSAs. In order to detect and reject false information due to an attack such as the D-LSA v.1 attack, it is necessary to have a source of trusted information to compare against. This could be collecting LSA information in a network that is not under attack or making use of layer 2 information which is not affected by these particular attacks.

5.3.3.2 Multi-Layer Authentication

Routing information is a claim about the connectivity and adjacencies between routers and/or routing devices. If adjacency information is also available from another source, it is possible to check routing information for inconsistencies. Information available from a lower layer is particularly desirable, as it is less subject to attack. A widely-used source of such information is the layer 2 protocol LLDP (or the similar Cisco proprietary CDP). Periodic requests can be sent out to gather information about LLDP to compare with OSPFv2 LSAs. The VRRC is designed to update the LLDP database at least once every default OSPFv2 update interval. This is illustrated in figure 5.2. The SNMP server gathers the LLDP data from each router in addition to any directly connected neighboring routers. For example, in a scenario with three routers, Router A <-> B <-> C, the SNMP server would collect the LLDP information from Router A, which would contain both LLDP information for itself and also LLDP information for Router B. In the case of Router B, router B will report its own LLDP information, in addition to Router A and Router C. This overlap in data helps provide additional information regarding a link's validity.

Multi-layer Authentication stores LLDP information in a database for comparison with LSAs. This will only occur if an LSA passes the first three LSA Inspection checks (e.g. if an LSA update is flagged as “Yes” for LSID, Router ID, and finally Advertising Router, then the LSA update is checked against the dataset from LLDP). Using the unique identifier of the MAC address and the RID, the same OSPFv2 LSA update can be compared to the data from layer 2's LLDP protocol. The result is data provided by layer 2 that can be compared to similar data about the existing routers/paths that exist from layer 3 devices. The benefit of leveraging layer 2 data is that layer 2 protocols are not vulnerable to attacks that specifically target layer 3 protocols and this information can be utilized to provide checks against potential rogue layer 3 devices.

5.3.4 Recover and Adapt

Routing updates contain the information necessary to establish connectivity and adjacencies between routers and/or routing devices, while router policies have the ability to create filters for
both ranges of address down to a specific host. Just as attackers use these updates to spread false information regarding the state of the network, defenders can use updates to recover from attacks by repopulating the routing tables, while simultaneously creating a rule to block the malicious host.

Forced Routing Updates (figure 5.2) is the initial recovery component implemented in the VRRC. Purging the OSPFv2 database on the virtual routers assures that all incorrect LSAs are removed. Since the three core routers are virtualized, their tables repopulate with one-another’s links almost instantly. Opposed to the physical links, which take slightly longer to re-populate. The command works by issuing a copy of each LSA currently in the OSPFv2 database with the age of every LSA set to MaxAge. The originator of each LSA is forced to generate a new LSA, updating information as needed. Since phantom routers and broken links cannot generate a new LSA, they and any other unnecessary LSAs are wiped from the routing table [Ger10]. The technique guarantees recovery from non-persistent attacks and some recovery from persistent attacks [Mer18]. Persistent attacks will be wiped temporarily from the table, however due to the nature of ‘persistent’ attacks, the router will become compromised again.

By automatically generating a set of policies on the virtual routers, such as through the use of Access Control Lists, the VRRC is able to adapt to the attack by generating a rule that will block the source of the attack while simultaneously alerting system administrators. While this does not prevent the attacker from changing sources and launching the attack again, it at least forces the attacker to change their initial strategy and increases the chances for network administrators and analysts to address the threat.

5.4 Experimental Evaluation

This section investigates the effectiveness of the VRRC in detecting persistent OSPFv2 attacks and the impact of deploying the VRRC on the network. Subsection 5.4.1 describes the lab conditions in which the experiments were run. Subsection 5.4.2 shows the effectiveness of the VRRC against attacks. Lastly, subsection 5.4.3 assesses the overhead of these methods.

5.4.1 Lab Setup

A development network was constructed using both physical and virtual routers. Both were commercially available production-class routers running the unchanged OSPFv2 protocol. The development network is illustrated in figure 5.3. It consisted of four physical routers, (three Cisco routers running version 15.2-4(M11) and a Juniper SRX running version 12.1x44-D35), and three virtual Juniper vSRX series routers (version 15.1x49-D70.3).

The virtual routers (vSRX1, vSRX2, vSRX3), SNMP server, and VRRC were installed on a Dell PowerEdge R330 server running Ubuntu Server 16.04 LTS. A virtual bridge was configured to permit
communication between the virtual routers, virtual SNMP server, and the three physical interfaces on the server. The physical interfaces had the following connections: eth0/0, eth0/1, eth0/2 were directly connected to R2, R1, and R4 respectively.

The attacks were stored on two laptops, labeled Attacker1 and Attacker2. When attacks were launched for an experiment, the laptops were plugged into a switch that was connected to one of the routers. The attacks were launched by different persons than were responsible for detection and recovery.

All of the attacks (Remote False Adjacency, D-LSA v.1 & 2, and the Partitioning Attack) were launched using the same network configurations under load. MD5 authentication was not implemented on OSPFv2 for two specific reasons. The first is that MD5 is vulnerable to key cracking, for which tools are readily available. [Ren10] Secondly, at least one of the attacks is unaffected by LSA authentication, even if more powerful cipher protocol suites are used [CN15].

\[\text{When testing the D-LSA v.2 attack, R2 and R3 were rolled back to version 15.2-4}\]
5.4.2 Results

Individual techniques were investigated for effectiveness before combining them with the VRRC. The first experiment implemented LSA Inspection, the second implemented Multi-Layer Authentication, and the third implemented Force Routing Updates. Lastly all three techniques were combined to demonstrate the overall effectiveness of the VRRC, which demonstrated the success of the VRRC in detecting all known persistent OSPFv2 attacks.

5.4.2.1 Effectiveness of LSA Inspection

In the first experiment, only the LSA Inspection portion of the VRRC was tested. Learning mode was initiated to collect uncorrupted LSAs, as would be the case in a new network. Then LSA Inspection was activated, and attacks were launched. False LSAs were consistently and successfully detected for the Remote False Adjacency, D-LSA v.1, and Partition Attacks, for both static and dynamic networks. The major limitation of LSA Inspection is the problem of distinguishing the D-LSA v.2 attack from normal addition of new routers and new interfaces in a dynamic network. Fortunately, Multi-Layer Authentication is able to address this problem.

5.4.2.2 Effectiveness of Multi-Layer Authentication

The second experiment tested just the Multi-Layer Authentication component of the VRRC. Multi-Layer Authentication makes use of layer 2 information about router interfaces which is dynamically collected and propagated to the SNMP server. This information allows the system to operate in a dynamic network topology by successfully differentiating between new legitimate links and new falsified links. Unfortunately, without the database generated by the LSA information, the data provided by Multi-Layer Authentication, in its current configuration, is only capable of determining whether or not a link is valid.

5.4.2.3 Effectiveness of Recovery and Adapt

The third experiment tested the initial recovery method of Forced Routing Updates and adapting to the attack through the generation of routing policies. While Forced Routing Updates was not capable of detecting any of the attacks, it did provide a degree of recovery. As discussed in section 5.3.4, removing false routes etc. was done by purging the routing tables. Generating a routing policy to block the false address allows the system to adapt to prevent the same source from persistently attacking. This clears the effects of false LSAs from the routing tables and prevents the same false route from being accepted into the routing tables again. In the case of ‘persistent’ attacks, it forces the attacker to re-launch their attack, thus providing system administrators with more opportunities to catch the attacker and creating more of a moving target.
5.4.2.4 Effectiveness of the Virtual Routing Redundancy Cluster

The last experiment tested the combination of all the above techniques in the Virtual Router Resilience Cluster. This combination was able to successfully detect and recover from the Remote False Adjacency, D-LSA v.1, D-LSA v.2, and the Partition attack (in addition to other non-persistent OSPFv2 attacks). The combination of using LSA Inspection to verify the legitimacy of the links, along with Multi-Layer Authentication to provide Layer 2 mapping for comparison of each RID and MAC address to each OSPFv2 LSA LSID and Source Router ID proved successful. Finally, recovery was
implemented by clearing the OSPFv2 database, thus forcing routing updates to assure that any false links are purged from the database. Simultaneously, the VRRC was able to adapt by generating routing policies to block the source of the attack. Which is particularly useful, since persistent attacks are particularly reliant on static targets.

Aside from the success of detecting the attacks, the VRRC was also successful in meeting our additional four criteria of not requiring any changes to OSPFv2 protocol, maintaining a dynamic network environment, backwards compatible with existing infrastructure, while generating only minimal overhead.

5.4.3 Network Statistics and Performance Measurements

The performance impact of using the proposed techniques was measured, in addition to effectiveness against attacks. The data collected included bandwidth (BW), latency (LAT), and packet-loss over time. Additionally, the overall resource usage of the VRRC was evaluated by measuring the impact on non-attack traffic for each of these mechanisms, under conditions of non-attacks.

The measurements were taken using a client / server configuration (The client was attached to R2 and server was attached to R4) running a modified version of qPerf to transmit TCP packets, while the server ran qPerf server to measure bandwidth and latency and TCPdump to measure packet loss. qPerf was modified to take multiple measurements back to back and in this configuration would take a complete measurements for latency and bandwidth every 4.013 seconds. 150 individual measurements were taken over the course of 10 minutes. Finally, the average was taken for every 15
individual measurements (roughly 60.195 seconds) for both latency and bandwidth.

*Figures 5.4, 5.5, and 5.6 compare bandwidth and latency in three distinct cases. The first case is termed “No Virtual Router” (NoVR); in this case, traffic did not encounter any virtual routers on the path (R4 → R3 → R2 in figure 5.3) The second case is termed “Base”; the path taken by traffic (R4 → vSRX1 → R2) included a virtual router, but none of the detection / recovery techniques were being executed. The last case is termed “Production” (Pr), in which the virtual router was executing the full capabilities of the VRRC.

The results were as follows. In the No Virtual Router case, the measured end-to-end bandwidth was 917 Mb/s, and the measured end-to-end latency was 250 microseconds. In the Base case, bandwidth was reduced from this level by .6%, and latency was increased by 18.4%.

Finally, for the Production case bandwidth was further reduced by 1.6% and latency increased by 21% from the Base case. Virtualization has an impact on packet latency. In none of the experiments was there significant packet loss (*table 5.2*).

<table>
<thead>
<tr>
<th></th>
<th>NoVr</th>
<th>Base</th>
<th>Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packetloss (%)</td>
<td>2.69%</td>
<td>3.28%</td>
<td>3.79%</td>
</tr>
</tbody>
</table>

In these experiments, the overall resource usage (Memory and CPU%) of the VRRC under load was very modest. The memory usage remained relatively constant at 30MB, while the CPU usage was .8%. CPU usage increased momentarily (< 1 s) when the VRRC state transitioned from/to Detection mode to/from Learning mode.

### 5.4.3.1 Deployment on Enterprise Backbone

With the assistance of the University’s Office of Information Technology Security and Compliance a VRRC was deployed on the backbone of the campus Network. The VRRC was directly connected to an OpenFlow switch that received traffic sent from a SPAN port between a pair of University backbone routers. The OpenFlow switch filtered all traffic except for OSPFv2 protocol messages (i.e., user traffic was never exposed or collected). The SPAN was configured to only allow one-way communication (i.e the VRRC could only receive traffic). That meant the VRRC did not force routing updates, originate LSAs, or serve as a designated router for a subnet. Note that the deployed VRRC did not receive LLDP information, preventing the use of Multi-Layer Authentication; only LSA Inspection was activated.

OSPFv2 messages were collected over a 10 week period. During deployment, the resource usage of the VRRC was relatively consistent with the previously-presented experimental results.
(Section 5.4.3). The memory usage (average) only increased slightly from 30MB on our Production Network to 33MB on the Enterprise Backbone. The CPU usage remained relatively the same at .85%. The ability to deploy the VRRC in a enterprise environment for an extended period of time has shown that the the VRRC is also scalable. Additional data continues to be collected.

5.5 Discussion and Limitations

Virtual Routing Resilience Cluster (VRRC) is successful in preventing attacks against OSPFv2. It stands apart from many other suggested techniques, in that it exploits existing network capabilities rather than requiring changes to the OSPFv2 protocol. The integration of virtual routers alongside physical routers results in a more resilient network. Additionally, the VRRC retains the OSPFv2 characteristic of maintaining a Dynamic Network Environment, is backwards compatible with existing infrastructure, and generates only minimal overhead; this makes VRRC practically more attractive than other proposed techniques [WW98; Wu99; Nak12; CN15]. While a component of the VRRC does have similarities to proposed mitigation techniques [Nak14], the key factor in VRRC’s strength is its novel approach of pulling data from other network layers, and its ability to recover from a large variety of attacks.

A potential drawback of the VRRC is with Forced Routing Updates. Purging the OSPFv2 database may increase latency, fraction of dropped packets, and time spent recalculating routing tables [Moy98b]. However, by having all of this take place on a virtualized platform, these effects are greatly reduced. Additionally, at present, the VRRC was only tested in a single-area OSPFv2 environment. For OSPFv2 deployments consisting of multiple areas it would be advisable to deploy one VRRC per-area.

5.6 Summary

This section presented a method of detecting and recovering from OSPFv2 attacks, using a Virtual Routing Resilience Cluster. Our results show that the VRRC performs well in practice, and poses an attractive broad-spectrum resilience mechanism against persistent OSPFv2 attacks. Future work will include additional deployment with full monitoring capabilities, and analysis of attack types and frequencies.

In the following chapter, we further advance the research of making OSPFv2 more resilient by demonstrating that implementation differences observed at the packet level between physical OSPFv2 routers can be used to differentiate between routers.
We establish that implementation differences observed at the packet level between physical OSPFv2 routers allow attackers to differentiate between router vendors. The widespread adoption of OSPFv2 as the interior gateway routing protocol of choice within enterprise and some service provider networks make the protocol an attractive target for malicious actors. While OSPFv2 is an open standards based protocol, the specification is non-exhaustive leaving some design decisions to the implementer.

Malicious actors may target OSPFv2 deployments by masquerading as routers, performing router-in-the-middle or other similar techniques. The ability to detect when one router is falsely claiming to be another router through the analysis of the routers fingerprint would be a powerful tool in the network operators arsenal. In this chapter, we analyze multiple packet captures taken from both physical and virtual instances of routers to confirm the presence of differences in OSPFv2 implementations and to characterize these differences. From this analysis we describe heuristics for predicting the OSPFv2 router based on identified characteristics and develop a software utility to conduct this analysis on packet captures. Ultimately we hope this analysis will provide valuable information enabling network operators to improve network resilience by making it more difficult for malicious actors to launch attacks against OSPFv2 networks.

The rest of this chapter is organized as follows: Section 6.2 discusses methods used to obtain OSPFv2 protocol data and methods of analysis to differentiate routers. Section 6.3 provides an overview of the experimental evaluation environment used to collect data. Section 6.4 discusses
characteristics we identified through experimental evaluation that allowed us to differentiate each router, while Section 6.5 highlights limitations and future work and Section 6.6 summarizes the chapter.

### 6.1 Introduction

The goal of this research is to assist network security specialists by characterizing the differences in OSPFv2 implementations across multiple routers in both physical and virtualized environments in order to perform router identification through fingerprinting. By identifying the nuances that differentiate an implementation of OSPFv2, we are able to provide network specialists with the information they may require to make their networks more resilient.

Although researchers have studied OSPFv2 protocol conformance in the past \[BJ08\] and have performed general black-box analysis of routing protocols \[Sos17\], the analysis was not focused on fingerprinting specific implementations to identify the router. Li et al. \[Li03\] focused primarily on analyzing the compliance of multiple OSPFv2 implementations with industry standards and Sosnovich et al. \[Sos17\] searched for deviations in the OSPFv2 implementations of Cisco routers. Sosnovich uses model compliance to compare two OSPFv2 routers, one Cisco and the other adhering specifically to the protocol in an effort to observe differences. While there has been previous work on fingerprinting software, most of it has been limited to Application layer \[Sma00; Mal00\]. Based on our analysis of the literature, this research presents a unique approach towards analyzing packet level characteristics of OSPFv2 implementations.

Our approach to identifying this problem and making the public more aware of the vulnerabilities it presents included consultation with multiple experts in the networking field from both academia and industry while analyzing these packet captures and conducting a systematic analysis of the data. The following research question was addressed:

- How can implementation differences observed at the packet level between physical OSPFv2 routers be used to differentiate between routers?

The contribution of this research is as follows:

- The identification of differences in default configuration parameters that are observable within packet captures can be used to assist in the identification of an OSPFv2 routers.

### 6.2 Methods

Measurements were taken in a lab environment modeled on the topology of NSF-net with routers utilizing the OSPFv2 routing protocol. OSPFv2 control plane packets including HELLO and LSAs
exchanged between routers in this realistic topology consisting of physical, virtual or a mix of physical and virtual routers were captured when links were added, removed and changed in cost. Two experts systematically analyzed the captures for differentiators in packets sent from different routers, and between virtual and physical routers. The observed differences were used to develop heuristics that attempt to identify the specific OSPFv2 router. These heuristics were then used to develop a software utility that can analyze packet captures to identify specific OSPFv2 router.

Once the packets were captured they were analyzed within a protocol dissector to examine similarities and differences between physical router implementations and physical vs virtual implementations. The analysis consisted of an examination of OSPFv2 protocol packet headers as well as an analysis of the timing of packets under various conditions described further within Section 6.3. Manual analysis within the protocol dissector revealed characteristics that were used to develop heuristics that assign a probability indicating the likelihood of each unique OSPFv2 router, identified by the OSPF router id. These heuristics were also implemented within a software analysis program to permit rapid analysis of additional captures. To assess the effectiveness of these heuristics, we applied them to packet captures that were collected from multiple links within a realistic network topology as well as to captures available on the Internet.

### 6.3 Experimental Evaluation

The experimental topology consisted of an amalgamation of both virtual and physical routers. For experimental purposes the physical routers consisted of Cisco model 1921 and Juniper model SRX240 routers. The virtual routers were virtual editions of their physical router counterparts and had consistent network operating system characteristics.

The 14-node NSF-Net topology was selected for these experiments for several reasons: NSF-Net is an accepted experimental topology and we wanted to demonstrate that our approach is not dependent on any particular topology configuration.

Experiment 1: Our first experiment was designed to look for differences between physical and virtual implementations of OSPFv2 across two different router implementations operating within the same network topology.

The following steps were repeated multiple times for each router instance. After all of the routers within the environment were activated, the network was left alone until it converged into a steady state. Next, the following sequence of changes were made to the OSPFv2 network (after each change time was allotted to permit the network to stabilize into a steady state):

1. Link addition
2. Link removal
3. Link cost change

The network was monitored and packets were captured from multiple locations during each experiment. The locations (links) chosen for capturing traffic were relative to the router having the changes made to it. For example, in Figure 6.2, router 1(WA) was a physical router in some test cases.
and was virtual router in other test cases. As a result, the links between R1 <-> R2, R1 <-> R3, R1 <-> R7, and R1 <-> R8 were monitored.

In this deployment, the link addition occurred between R1 <-> R7 and the link removal occurred between R1 <-> R8 while the link that was subject to a cost change was R1 <-> R3. For the change in cost, the interface of R1 and R3 (both interface ge0/1) are set to a default value of 1. When the cost (metric) is changed for the links, R1 interface is assigned the cost of 100 while the interface on R3 is assigned the cost of 1000. By setting this cost, we are able to manipulate the shortest path calculation in such a way that the link connecting R1 <-> R3 is never utilized in the path selection.

This process was carried out for both physical R1 and then again for virtual R1.

6.4 Results

Our findings from the above testing scenarios are characterized as follows:

6.4.1 Timing Analysis

Juniper has a configurable LSA update interval [Net18a] and by default JunOS utilizes an update interval of fifty minutes while Cisco by default uses the RFC2328 [Moy98b] specified 30 minute update interval and has a more complex optional flood reduction mechanism [Sys18b]. While different LSA update intervals were observed across router vendors due to differences in default configuration parameters, no differences were observed across physical and virtual platforms from the same vendor as the default configuration parameters were consistent between physical and virtual platforms from the same vendor.

6.4.2 Hello Analysis

The OSPFv2 hello packet analysis revealed another configurable parameter with different default values across the vendors analyzed: The default designated router priority used by Cisco has a value of 1 [Cis18] while the default designated router priority used by Juniper has a value of 128 [Net18b]. While different designated router priorities were observed across vendors due differences in default configuration parameters, no differences were observed across physical and virtual platforms from the same vendor as the default configuration parameters were consistent between physical and virtual platforms from the same vendor.

6.4.3 LSA Analysis

One of the most interesting findings occurred while examining the contents of LSA updates within the protocol dissector. It seems as if the links contained within an LSA update originated by a Juniper
router are sorted in ascending order as shown in the excerpt below:

Juniper (ascending order): Neighbor Network-ID: 10.0.3.10, Interface Address: 10.0.3.10 Stub Network: 10.11.41.0, Mask: 255.255.255.0 Stub Network: 10.11.42.0, Mask: 255.255.255.0

With Cisco routers the links are sometimes in descending order as shown in the excerpt below:

Cisco (descending order): Stub Network: 10.11.32.0, Mask: 255.255.255.0 Stub Network: 10.11.31.0, Mask: 255.255.255.0 Neighbor Network-ID: 10.0.3.10, Interface Address: 10.0.3.13

Although it should be noted that there are other times where links originated by a Cisco router are in arbitrary order. This finding is significant in that this appears to be a vendor specific implementation choice that does not have configurable parameters permitting the administrator to alter the order of the prefixes advertised within LSA.

### 6.4.4 Heuristic Construction

Based on the findings from the experimental analysis outline above, the following heuristics were developed to assign each OSPF router id identified within a packet capture a probability of being a Cisco or Juniper device. Given that these are heuristics, the probabilities assigned to the rules may be adjusted as needed:

1. In an OSPF hello packet, if the designated router priority = 1 increase Cisco probability by 0.1 and decrease Juniper probability by 0.1.
2. In an OSPF hello packet, if the designated router priority = 128 increase Juniper probability by 0.1 and decrease Cisco probability by 0.1.
3. If an OSPF LSA update packet of Type 1 (Router LSA) is encountered and the router id matches the advertising router:
   (a) If the links IDs are in descending order increase Cisco probability by 0.5 and decrease Juniper probability by 0.5.
   (b) If the link IDs are in ascending order increase Juniper probability by 0.6 and decrease Cisco probability by 0.5.
   (c) If the link IDs are in mixed order do not change the Cisco probability and decrease the Juniper probability by 0.6.
4. Reset any probability values that were decremented to be less than zero to zero.
6.4.5 Performance Results

An analysis utility was developed in Python [Ros18] that leverages an open source packet parsing and construction library called Scapy [Bio18] as well as an open source OSPF extension to Scapy [(2016)] to permit parsing of the packet captures from the experiential evaluation which were stored in libpcap format. The analysis utility can read one or more packet capture files at a time and will identify OSPF routers within the capture by router id and will then assign a probability of each router id as being a Cisco or Juniper device by leveraging the heuristics outlined above. The results of the analysis software are in a comma separated value format that can be opened with a variety of tools for further analysis.

Running the software analysis utility against all packet captures collected during the experimental evaluation described in Section 6.3 provided a vendor probability for each router id observed within the packet captures. The probability derived from the heuristics are compared to the actual vendor of the device as highlighted within Figure 6.3.

![Software Analysis Heuristic Results](image)

**Figure 6.3 Software Analysis Heuristic Results**

Within Figure 6.3, the Cisco routers are shown on the left and the Juniper routers are shown on the right. While none of the heuristics predict the router vendor with 100% confidence (as our heuristics
do not contain a complete set of indicators that sum to 100%), none of the predictions were wrong and all of the heuristic results matched the actual routers used within the experiment.

To conduct additional analysis, we collected additional packet captures from repositories on the Internet [Str09; Cap06] and ran the software analysis tool against these captures. The analysis software predicted that the routers were Cisco which was confirmed in [Cap06] by examining the OUI of the MAC address and assumed in [Str09] based on other indicators on the download repository.

6.5 Limitations and Future Work

The OSPFv2 routers analyzed in this work were limited to physical and virtual models from both Cisco and Juniper. However, we felt this sampling size was a sufficient starting point. Because if no initial differences were observed in such a small sampling, then a router’s OSPFv2 fingerprint would not be useful in detecting malicious actors. With the appropriate access to packet captures from others routers the work could be extended to provide potential heuristics to identify a larger set of both commercial and open source OSPFv2 implementations such as Quagga. The work could also be extended to perform similar analysis on OSPFv3 which can support multiple address families including IPv4 and IPv6.

The software analysis package could be extended to analyze the LSA update interval as a contributor in the heuristic prediction of the router. In addition, while we tried to focus our analysis exclusively on the OSPFv2 protocol headers additional header analysis could lead to heuristics with even greater accuracy - as an example, the Layer 2 headers could be examined to determine the interface manufacturer by OUI lookup. Similarly, the presence of any link-layer discovery protocols such as the Cisco Discovery Protocol (CDP) [Sys18a] or Link Layer Discovery Protocol (LLDP) [Con02] could be examined.

The specific heuristics developed as a part of this research successfully provided an indication of the router in the experiments conducted but it should be noted that routers could change default configuration parameters or the protocol implementation which may require an update to the specified heuristics. More importantly, this research illustrates that it is possible to fingerprint an OSPFv2 router regardless of the specific heuristic used at the protocol level.

6.6 Summary

The ability to fingerprint routers without prior knowledge of network infrastructure is a significant risk to network operations since bad actors may be able to leverage this information to craft attacks against specific vulnerabilities of specific routers models, target OSPF deployments by masquerading as routers, perform router-in-the-middle or other similar techniques. By identifying the nuances that differentiate an implementation of OSPFv2 between routers, we are able to provide network
specialists with the information they may require to make their networks more secure. This research highlights the need for future protocol designers to keep security considerations in mind while developing protocol specifications as it seems that any aspects of the protocol that are left to the implementer are candidates for fingerprinting or potential interoperability issues.

In the following chapter, we make OSPFv2 even more resilient to persistent OSPFv2 attacks, through the use of an innovative prevention method that utilizes time as a means of authenticating LSA updates.
In this chapter we present the LSA Synchronized Time Controller (LSTC). The LSTC makes use of a deterministically pseudorandom number generator designed to calculate time intervals for LSA Updates, thus randomizing the LSA update times on all routers and increasing the difficulty in gleaning the necessary information to launch a targeted network attack. We demonstrate that the LSTC makes the network more resilient against all known persistent OSPFv2 attacks and that the concepts are feasible to implement without resorting to the complexity of key management, and without modifying the routing protocol.

7.1 Introduction

The widespread adoption of OSPFv2 as the interior gateway routing protocol of choice within enterprise and some service providers dictates an ever increasing need to make networks that implement OSPFv2 more resilient to malicious attacks. Despite how robust the distributed operation of routing is to normal failures, routing is vulnerable to deliberate attack (E.g. eavesdropping and man-in-the-middle) [Nak12; Jon; Jou00]. Malicious attacks intentionally cause routers to fail at a highly accelerated rate, or to propagate incorrect or inconsistent information throughout the network [GoI11].
Spoofing OSPFv2 LSA Updates provides a way for an attacker to compromise the OSPFv2 routing table by circumventing the OSPFv2 fight back mechanism. What makes spoofing easy for an attacker is the predictability of the OSPFv2 update times, either per the RFC 2328 or vendor implementation. This predictability of the LSA update times makes it easy for attackers to appropriately craft their packets to spoof the updates to spoof legitimate updates.

In this chapter we present the LSA Synchronized Time Controller (LSTC). The LSTC makes use of a deterministic pseudorandom number generator to calculate time intervals for LSA Updates, effectively randomizing the LSA update times on all routers and increasing the difficulty in gleaning the necessary information to launch a targeted attack against OSPFv2 LSA Updates.

Our contribution is thus two fold:

1. Demonstrate that is it possible to make modern routers in distribution generate LSAs at specific instance, overriding their periodic nature, without extensive modification.

2. Demonstrate that with such a synchronized pseudorandom LSA distribution, we can nullify attacks against OSPFv2 that depend on sending spoof LSAs.

We demonstrated that in order to keep false negatives and false positives below 10% the window size (the amount of time the LSTC allows a router to transmit and receive LSA information) should be between 750ms and 1s. In this time range we show that at 750ms no attacks (false positives) were detected, with a false negative reading of roughly 11% and at 1s, the false positives were roughly 10% with a false negative reading of .02%.

The chapter is organized as follows. Section 7.2 briefly summarizes the operation of OSPFv2, reviews the known attacks against OSPFv2, and summarizes prior work on defenses against such attacks. Section 7.3 proposes a new approach to preventing and detecting OSPFv2 attacks through the use of a centralized time-clock. Section 7.4 describes an implementation of this approach, and an evaluation of its effectiveness and costs. Section 7.5 discusses the benefits and limitations of the proposed approach, and section 7.6 summarizes the chapter.

### 7.2 Background

Several studies have been conducted on resilient networks, e.g. [Smi11], however most of this prior work does not address malicious threats. Additionally, most solutions involve either making significant changes to the network infrastructure or to the protocols themselves. In the case of attacks against OSPFv2, specifically exploits against LSA, several solutions were proposed by [WW98]. One solution discussed suggests the use of a secure wrapper with different levels of checking gates for input and output between the secure wrapper and the protected router. The secure wrapper examines all data, using a security policy database to determine the status of these checking gates.
[Wu99] introduces a scalable IDS incorporating a ‘detection’ module that performs statistical and protocol-based intrusion analysis and allows the user to remotely modify the ‘prevention’ and ‘detection’ modules based on the detection information. [Nak12] discusses two potential measures that could be taken to prevent the remote false adjacency and disguised LSA. The remote false adjacency could be thwarted if the AS ensures that different links use independent secret keys. This solution requires complex and error prone key management practices. In the case of the D-LSA v.1 attack, it was proposed that one could mitigate the predictability of the LSAs by advertising false links with random values to randomize the LSA checksum. Unfortunately, this technique would also be impractical for deployment in real-world applications because it would ultimately result in increased routing table sizes.

7.3 Timing as Secure Channel

Timing can be used by a sophisticated attacker against OSPF, by targeting spoof LSAs at times when they would be most effective. Our idea is based on closing that door, and also using timing itself as a mechanism to secure the LSAs instead.

In TDM oriented protocols, such as frame relay or NG-SONET, what time slot some data arrives in can itself act as addressing or labeling information for the data. As it is in GMPLS, where a time slot can be a label. We have the idea of doing this same thing for OSPFv2, by tacitly designating certain short windows of time to indicate (be valid for) legitimate LSAs. Implicitly, any LSA at any other time is assumed to be non-legitimate, and either recognized as an attack, or ignored.

Unfortunately, changing the timings of the LSA updates would constitute changing the protocol, which would not be realistic. However, instead of changing the protocol, we are merely controlling the operation of the protocol, by allowing LSA updates to only go out and be received at certain windows, by using the CLI (or other management channel) that any router must have.

However, if these timeslots were to remain regular, then even a marginally skilled attacker would be able to identify these timeslots after a period of time. As such, a random sequence is required. But then to coordinate the routers, we would need a control channel - which would reintroduce the attack surface.

Our solution is the use of a deterministic pseudorandom sequence, which can be generated in hardware by a separate security device (“dongle”). This device can be replicated, and each device will generate precisely the same sequence of intervals. A number of them can then be distributed throughout the network (one on each set of existing routers), allowing the routers to be automatically controlled to open and close the LSA windows in sync with each other.
7.3.1 Detailed Methodology and Test Environment

For the purposes of our research, we will deploy the LSA Synchronized Timing Controller (LSTC) on a development network without the need for a separate trusted channel to send update times to each of the routers from a central location. Instead, a Random Time Interval Generator is attached to each router. In practice, this can be done through the hardware dongle described previously. The LSTC does require precise synchronization of the clocks on all routers and Random Time Interval Generators to the same time. For example, in our experiments all devices were synchronized through NTP.

Throughout the network, each OSPFv2 routing device will have a Random Time Interval Generator. The Random Time Interval Generator is designed to deterministically generate pseudorandom numbers used to calculate time intervals for LSA Updates (Every Random Time Interval Generator utilizes the exact same deterministic pseudorandom number generator and seed), and to forcefully trigger the LSA Updates. The Random Time Interval Generator over-rides the default LSA updates times (e.g. Cisco every 30min and Juniper every 50min) found on all vendors’ routers.

The Random Time Interval Generator calculates the random time intervals as follows, given the range of possible update interval times: 9min.00sec.00ms ←→ 59min.00sec.000ms where there minimum time is \( t_{\text{min}} \) and the maximum time is \( t_{\text{max}} \):

\[
\begin{align*}
  t_{\text{min}} &= 9\text{min}.00\text{sec}.000\text{ms} \\
  t_{\text{max}} &= 59\text{min}.00\text{sec}.000\text{ms}
\end{align*}
\]

the total time \( t_{\text{tot}} \) is the difference between the \( t_{\text{max}} - t_{\text{min}} = t_{\text{tot}} \):

\[ t_{\text{tot}} = 3,000,000\text{ms} \]

A random time \( r \) is calculated such that

\[ r = \text{gen\_random}(0, t_{\text{tot}}) \]

where \( \text{gen\_random} \) can be any secure pseudorandom number generator (as long as the same deterministic pseudorandom number generator is used on all Random Time Interval Generators).

Finally, the random LSA Update time is equal to the minimum time plus the random time.

\[ \text{LSA Update Time} = t_{\text{min}} + r \]

Once a random time interval has been selected, the Random Time Interval Generator will begin a countdown timer based on the randomly chosen time. Once the countdown timer expires, the Random Time Interval Generator will trigger an LSA Update on the router. The Random Time
Interval Generator then waits for a set amount of time (window of time), expecting to receive an LSA Update. If the LSA arrives during this window, it will be accepted into the routing table. If an LSA Update arrives outside this window, the LSA Update will be ignored and a system alert will be triggered. Additionally, if no LSA arrives for a period of three countdowns, a system alert will be triggered. The Random Time Interval Generator will then choose another random integer and repeat the process as seen in Figure 7.1.

Example: All of the Routers, Random Time Interval Generators are synchronized using NTP to the Time Server. R1 is a Cisco router and by default is sending an LSA update every 30 min per the Cisco implementation of RFC2328 as seen in Figure 7.2. R1 has an external hardware device, known as the Random Time Interval Generator attached to it. The Random Time Interval Generator is designed to work with any router vendor and over-ride the default LSA updates times. The Random Time Interval Generator using a deterministic pseudorandom number generator (sharing the same seed across all Random Time Interval Generators) to generate a random integer (time). In this case, we will use the example of 25m:25s.025ms. The Random Time Interval Generator will initiate a countdown timer based on the chosen integer (e.g. 25m:25s.025ms). After the countdown timer...
expires, the Random Time Interval Generator attached to R1 will trigger an LSA Update on R1. The Random Time Interval Generator will not only trigger an LSA Update but it will also listen to receive LSA Updates from other routers within the network within the specified window of time. If the triggered LSA Update (on R1) is received (by R2) in the expected window of time, the Random Time Interval Generator will allow the LSA Update to be accepted into the routing table. However, if the LSA Update is received by R2 outside of that window, it will ignore and block the update.

### 7.3.2 Using Time Triggered LSA Updates to Prevent Spoofed LSAs

A deterministically pseudorandomized Triggered LSA Updates has the ability to prevent spoofed LSA updates. Spoofed LSA Updates rely on the ability of the attacker to beat the OSPFv2 fight back mechanism in an effort to make the victim router believe that the spoofed LSA Update is legitimate because it ‘arrived’ before the legitimate LSA Update. As a result, the victim router will ignore the genuine LSA Update, and accept the spoofed LSA update because the victim router believes the ‘spoofed’ LSA update is indeed the legitimate update containing the correct routing information.

---

**Figure 7.2** Lab Topology

---
The use of deterministically pseudorandomized triggered LSA Updates makes it even more difficult to predict when the next LSA update will occur and thus craft the ‘spoofed’ packet accordingly.

By controlling the time LSA Updates are triggered and the time they are allowed to be received, it is possible (within a certain margin of error), to prevent spoofed LSA Updates.

7.4 Experimental Evaluation

This section investigates the overall effectiveness of the LSTC. Specifically the Random Time Interval Generator at generating LSA Updates at specific instances of time and preventing and detecting falsified LSA Updates. In subsection 7.4.1, we begin by describing the lab conditions in which the experiments were run. Next we measure the experimental results gathered from the Random Time Interval Generator in subsection 7.4.2. These results evaluated the effectiveness of controlling the LSA Update times on the each router in addition to comparing the different window sizes in which the triggered LSA Updates could be both sent and received. Subsection 7.4.3 evaluates the success of a naive attacker against a sophisticated attacker attempting to inject spoofed LSAs into the network. Finally, in subsection 7.4.4, we discuss the overall measurements (False Positives, False Negatives, and Resource Usage).

7.4.1 Lab Setup

A development network was constructed using virtual routers located on a GNS3 [Gns] server hosted on ESXi. The images for the virtual routers are commercially available production-class routers running the unchanged OSPFv2 protocol. The development network is illustrated in Figure 7.2. It consisted of five virtual routers, (three Cisco routers running version 15.2-4(M11) and two Juniper virtual routers running ‘firefly’.

Both of the major vendors were used in the construction of the development network. The reason for using both vendors was to emphasize that the attack exploits the LSA Update process of OSPFv2, regardless of vendor OS or vendor implementation (e.g. Cisco issues OSPFv2 LSA Updates every default 30min per RFC-2328, while Juniper issues OSPFv2 LSA Updates every default 50 minutes).

The three hosts connected to the network, as seen in Figure 7.2 are virtual machines hosted on an ESXi server.

Host 1 contained the Random Time Interval Generator for each router. To emulate the way the Random Time Interval Generator would trigger an OSPFv2 Update, Host 1 had a direct console connection to each router in the topology (the same as simply plugging in a different Random Time Interval Generator to every router). Host 1 (where the Random Time Interval Generator was installed) generated a deterministic pseudorandom number for each of the attached routers, then initiate a countdown. This countdown time was the same for each router. Once the countdown
timer expired, the Random Time Interval Generator on Host 1 would send a signal to each router to open a window for a period of time, trigger an LSA Update, listen for incoming LSA Updates, then close the window to prevent further LSA Updates.

Host 2, contained the “Attack” VM, which was used to launch attacks against the network. The ‘attack’ was only designed to spoof an LSA Update and inject the falsified into the routers database. This LSA Spoof technique had been used in our previous work [Mer18] as the initial stage for launching the partition attack. Host 2 was connected to R3. From this direct connection host 2 would attempt to spoof LSA Updates on R3. Host 2 was designed to launch two specific types of ‘attacks’. Attack 1, was a random or ‘naive’ LSA attack. The naive LSA attack would randomly choose times to try to falsify an LSA Update. Attack 2 was a ‘sophisticated’ LSA attack. The sophisticated LSA attack would wait until a LSA Update on the network was observed, then it would launch the attack.

Host 3 contained the Time Server (NTP server). The NTP server ensured that consistent time was maintained on the Random Time Interval Generator, and all of the attached routers.

Each experiment was conducted under the same settings. The first experiment evaluated the feasibility of generating LSAs at specific instances of time without any modification to the OSPFv2 protocol itself or extensive modifications to the physical routing hardware. The initial experiment was carried out by using the Random Time Interval Generator to generate a deterministic pseudorandom number on all five connected routers as seen in Figure 7.2. The second experiment evaluated how effective a reduced window of time would be in preventing both naive and sophisticated persistent OSPFv2 attacks.

### 7.4.2 Random Time Interval Generator

We begin by comparing when LSA updates were received from routers associated with Random Time Interval Generators to the window where they would be expected to be received. The ‘t’ value represents the various size (time) for each window. A range of windows (250ms, 500ms, 750ms, 1s, and 2s) were examined to determine how many valid updates were received and how many valid would be rejected. We evaluate the tradeoffs of each window size in Section 7.4.4.

Table 7.1 displays the time the LSA Updates were triggered by the Random Time Interval Generator for each of the five routers. As we can see, the time window 250ms was not long enough to allow for an update to trigger. However, for each of the other four ‘t’ values, LSA Updates are generated. At 500ms and 750ms, it can be observed that during that particular window capture, only one LSA Update was successfully triggered and allowed to propagate. As the window size increased, we see that all five routers had LSA Updates triggered.

Table 7.2 displays the time at which the triggered LSA Updates from Table 7.1 were successfully received during the same selected time window from which they were triggered. (Note: Since, 250ms was not long enough to trigger and send an update, it subsequently did not receive anything and
Table 7.1 Triggered LSA Updates

<table>
<thead>
<tr>
<th>Window</th>
<th>Update Time</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>250ms</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>500ms</td>
<td>09:45:05.946</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>09:45:05.496</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 7.2 Receiving LSA Updates

<table>
<thead>
<tr>
<th>Window</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
</tr>
</thead>
<tbody>
<tr>
<td>500ms</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>11:02:26.542 R4</td>
<td>11:02:26.531 R3</td>
<td>R2</td>
<td>R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:27:54.662 R5</td>
<td>11:27:53.624 R4</td>
<td>R2</td>
<td>R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:27:54.682 R4</td>
<td>11:27:53.624 R2</td>
<td>R2</td>
<td>R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:27:55.142 R7</td>
<td>11:27:53.624 R5</td>
<td>R5</td>
<td>R5</td>
<td></td>
</tr>
</tbody>
</table>

therefore is not included in this table).

The following example explains how the two tables are connected. When the window (‘t’ value) was set to 2 second, after the random countdown timer expired, the window was set to open 11:27:53.397ms. During this time one can observe the LSA Updates that were sent in Table 7.1 and the LSA Updates that were received in Table 7.2. Each window was tested 30 times, to allow for variation of any underlying timing issues or other random factors. An ‘X’ indicates that an LSA Update was not Triggered (or received for Table 7.2) at that particular time. It was observed that when the window was under 1s, not every router would have time to finish generating the LSA Update and send it out in the narrow window of time.

7.4.3 Preventing Spoofed LSAs

Once we determined that it was possible to trigger the randomized LSA Updates for the network, it was important to evaluate how much more secure this would make the network. We first began by determining the probability that a naive attacker could “randomly” launch an attack that would be capable of finding it mark on the window size. For example, if the window size was set to 1s, the naive attacker, attempting to randomly inject a packet, has roughly a 1 in 3,000 chance of succeeding. This
attack was not successful, and statistically would only be successful if the attacker left the attack running for an hour. However, this would not be practical as the system admin could easily detect the spamming from of a single address.

The second experiment assumed a sophisticated attacker. In order for this attack to work, the attack had to monitor the network to determine when the first packet in an OSPFv2 LSA Update was sent. It was verified that after 30 runs, with a 2 second window a ‘skilled’ attacker could compromise a directly connect router roughly 26% of the time. When the ‘t’ value was 1 second, a skilled attacker could compromise a directly connected router roughly 10% of the time.

7.4.4 Optimization & Cost

For the purpose of the experiment/study, we consider the “optimal” solution one that reduces both False Negatives and False Positives. False Positives are the legitimate LSA updates that were received outside of the time window as examined in Section 7.4.2. False Negatives are defined as attacks that were received during the randomly selected time window as examined in Section 7.4.3. It was observed that over the course of 30 runs for each ‘t’ (window size) value the following information.

- When ‘t’ was 500ms, 10 of the 36 (or 27%) legitimate LSA Updates were received outside the window.
- At 750ms, 8 of the 73 (or around 11%) legitimate LSA Updates were received out of the window. At 1s, only 4 (or 2%) of the 150 legitimate LSA Updates were outside the window. By Comparison, we see that at 2s, 8 attacks were successful. While at 1s, only 3 attacks were successful. The number of False Negatives (rejected legitimate updates) and False Positives (successful attacks) for each window can be seen in the Figure 7.3.

This leads us to conclude that the ideal window size for the above network would lie somewhere between 750ms and 1s. This would ensure an extremely low ratio of False Negatives to False Positives.

Finally, in a random sampling of 30 runs, the Random Time Interval Generator was found to only use (on average) .17% of the CPU and around 13MB of memory.

7.5 Discussion and Limitations

The LSTC is successful in preventing attacks against OSPFv2. It stands apart from many other suggested techniques, in that it exploits existing network capabilities rather than requiring changes to the OSPFv2 protocol. The integration of a deterministic pseudorandom number generator to randomly trigger LSA Updates for each directly connected router proved successful in make the network more resilient to attacks designed to exploit the OSPFv2 LSA Update process.

Additionally, the LSTC retains the OSPFv2 characteristic of maintaining a Dynamic Network Environment, is backwards compatible with existing infrastructure, and generates only minimal overhead; this makes LSTC practically more attractive than other proposed techniques [WW98;
Finally, as always, the adversary does not stand still. As networks adopt the LSTC to make their networks more resilient, it is likely that persistent attacks will also grow more sophisticated. This further underscores the fact that more work is needed on resilience. The goals of such work must be reduce overhead and improve the flexibility of making current routing protocols more resilient to intelligent adversaries, while attempting to stay one step ahead of an increasingly sophisticated persistent attacker.

7.6 Summary

This chapter presented a method of detecting and recovering from OSPFv2 attacks, using a LSA Synchronized Time Controller. Our results show that the LSTC performs well in practice, and poses an attractive broad-spectrum resilience mechanism against persistent OSPFv2 attacks. Future work will include additional deployment with full monitoring capabilities, and analysis of attack types and frequencies. In the next chapter we summarize and conclude our research.
The current deployment of OSPFv2 was standardized almost two decades ago and still remains the primary ‘interior’ routing protocol around the world. In Chapter 4, we discuss the need for added resilience to OSPFv2 networks by reviewing the five primary recovery techniques (Restore, Rollforward, Reinitialize, Reconfigure, and Replace) and then experimentally evaluated how successful each of these techniques were against modern persistent OSPFv2 attacks. It was determined that only two (Reconfigure and Replace) of the five primary recovery techniques were successful in recovering from a persistent OSPFv2 Partitioning and DoS attacks.

Chapter 5 further synthesized recovery and resilience to form the Virtual Router Redundancy Cluster (VRRC), a system designed to seamlessly integrate with any existing OSPFv2 network, making the network more resilient to all known persistent OSPFv2 attacks in addition to providing a rudimentary network recovery mechanism. The VRRC proved successful in detecting all known persistent OSPFv2 attacks (e.g. Remote False Adjacency, Disguised LSA v.1, Disguised LSA v.2, and Partitioning). In the wake of the attack, the VRRC was also able to successfully recover network functionality (in this case purging the falsified links).

Chapter 6 builds upon these previous advances, by further exploring the issue of when a resilient OSPFv2 system should trigger auto-recovery. Routers can “fingerprint” each other after the network is initially brought up in a trusted environment or are supplied with such fingerprints obtained by testing in a trusted environment, like ISP’s NOC or deployment lab. They watch each other, and if some fingerprint changes, it may indicate that somebody is spoofing, or the router is compromised -
then they can trigger one of the recovery mechanisms. By analyzing OSPFv2 packets, we were able to successfully fingerprint two different routers. The success of this initial step will allow us to further explore fingerprinting an even larger selection of routers in the hope that it will be possible to detect when an attacker is spoofing a router by viewing attackers spoofed routers fingerprint.

Finally, Chapter 7 provides a novel means of preventing spoofed LSAs by using time as a means of authenticating LSA updates. Routers can agree, in advance and out of band, on a very specific sequence of intervals at which they will generate LSAs, and treat any LSA that is generated at any other time as attacks - therefore simply ignoring them (and possibly taking other recovery or retaliation mechanisms). Ideally the intervals would be a deterministically pseudo-random sequence, so that an observer cannot discern any pattern even if they observe every LSA transmitted by every router, but each router precisely agrees on the intervals. In principle, this can be achieved by a set of identical hardware dongles that are attached to each of the routers before deployment. In practice, one can also use a secondary control-plane network, if available and sufficiently secured, to distribute such time signals. By using time we were successfully able to prevent spoofed LSA updates. Through the analysis of false positives and false negatives gathered from experiments of different time windows, we determined that it is possible to have an LSA update windows between 750ms and 1000ms that should be capable of preventing any spoofed LSA updates. These positive initial results have emboldened us to further develop the LSTC outside of a virtual environment. In addition, we plan to make the LSTC provide even more security to LSA updates by having each dongle hold the seed for every other dongle, that way each dongle can have a different randomized countdown time for triggering an LSA update.

In this dissertation we focused on several subcategories within the umbrella of resilience (prevent, detect, and recover). It is our hope that the research provided within this dissertation will: 1. Further advance the knowledge in the area of network resilience, 2. Stimulate future research into recovery, and 3. Foster further research into making routing protocols such as OSPFv2 more resilient to attackers.


[Cap06] Captures, W. C. S. hsrp-and-ospf-in-LAN (libpcap) HSRP state changes and OSPF LSAs sent during link up/down/up. 2006.


APPENDICES
APPENDIX

A

NETWORK ATTACKS: DIAGRAMS AND WEEKLY SUMMARY

A.1 Diagrams of Attacks Against OSPF
Figure A.1 Diagram of LSID Attack.

**LSID Attack**

2. attacker sends back modified LSA

1c. victim continuously seg faults if bad LSA not purged

By Russell Meredith & Natalie Landsberg

1b. Router Database
   i. ls_id != router ID
   ii. index pointer to database is null, router tries to access ls_age and seg faults
MaxAge Attack

1. Attacker receives an LSA from the victim router and sets the age to MaxAge (1 hour). \texttt{ls\_age = MaxAge}
2. Attacker re-injects the LSA into the system.

![Diagram of MaxAge Attack](image)

Figure A.2 Diagram of MaxAge Attack
Figure A.3 Diagram of Max-Seq Number Attack.

1a. attacker intercepts LSA

1b. Isa_seq = MaxSeqNumber (0x7FFFFFFF)

2a. attacker injects modified LSA

2b. link cost significantly increases OR attacker controls database for up to 1 hr (MaxAge)

By Russell Meredith & Natalie Landsberg
Figure A.4 Diagram of Sequence++ Attack.

Seq++ Attack

1. Attacker receives an LSA from the victim router and increases the LSA sequence number by 1.
   - must also re-compute both LSA & OSPF checksums
2. Attacker re-injects the LSA into the system.
Remote False Adjacency

1. Send false messages to learn intervals for routers (HelloInterval, RouterDeadInterval, etc.)

2. Inject LSA with a non-existent router
   2a. No fight back triggered since router does not exist

3. Can be used to:
   a) blackhole: redirect traffic into the abyss
   b) redirect traffic between two routers by saying the link cost is smaller

Figure A.5 Diagram of Remote False Adjacency Attack.
Figure A.6 Diagram of Disguised LSA v.1.

1. Spoofed LSA of Hyperion ("Trigger" LSA)
   1a. Hyperion rejects duplicate advertisement

2. Simultaneously, send Disguised LSA of
   Hyperion to Vladof containing future fight back LSA of Hyperion

3. Hyperion sends fight back to Vladof

4. Vladof rejects legitimate update, using more 'recent' (disguised) LSA

5. Vladof re-floods the disguised LSA to Hyperion

6. Hyperion will view Vladof's update as the one it sent in step 3 and will not fight back

By Russell Meredith
Figure A.7 Diagram of Disguised LSA v.2.

1. Trigger "fight-back" to learn:
   Link State ID, Advertising Router, & Sequence #

2. Advertise LSA where:
   - Link State ID = Vladof Router ID
   - Advertising Router != ID of Vladof
   - Sequence # > Vladof
Attacker sends an LSA to every one of the victim’s neighbors. Each LSA indicates that the destination router has no link to the victim, thereby creating different topologies in each router’s database and forwarding loops.

Figure A.8 Diagram of Partition Attack.
APPENDIX

B

NETWORK ATTACK GENERATOR

B.1 Attack Generator

The objective of ‘Red Team’ was to develop a program, capable of safely launching and testing a variety of different real-world network attacks. The challenge was taking a variety of attacks which were built using different programming languages and toolkits and making it so they could all be launched from a central location and in a scriptable manner, all the while ensuring proper monitoring, containment, and guaranteeing the attack is launched against a specific target.

The system, nicknamed Excalibur, primarily consists of three parts: the launcher, the individual attack scripts, and a repository for the exploits themselves. The launcher, which is essentially an argument parsing system is designed to gather command-line input such as a network interface, target IP, attack number, etc. The input is first validated and then passed to either a single or a series of ‘attack scripts’ which simply re-organize the input as given by the launcher into a form that fits a given exploit’s expected input and then execute the actual exploit. The launcher keeps track of available exploits by searching through a repository folder for attack launching scripts with a specific naming convention. The attack scripts add a layer of abstraction so that exploits produced using any language or system can be launched in a predictable and easily scriptable manner, while also allowing for the storage of optimal attack configurations (for example, an attack script could
always launch an exploit with specific, tuned, command-line arguments). These scripts consist of no more than a few lines of code, as they simply re-organize input and/or gather new information if necessary, and then execute the command required to launch an attack. This setup allows new attacks to be added to the repository extremely quickly and without requiring modifications to be made to the launcher or exploits. Finally, the repo of exploits consists of a variety of network attacks, all of which were all discovered in publicly accessible locations online. Many are open source or have public documentation. Examples include a CDP table flooding attack, TCP SYN flood scripts, DHCP starvation attacks, OSPF route injection attacks, OSPF table flood attacks, router MAC table overflow attacks and others.

These attacks were selected to create a varied real-world suite of network attacks in order to adequately test and observe their behavior when launched against layer 3 devices in a real-world network. Attacks targeting solely very specific systems (for example, zero-days effective against a single router firmware version) were not selected; rather, those that could have a major impact on a variety of systems became the focus of this repository. Overall, this system allows for many different network attacks to be run against a targeted system in a contained and easily scriptable way using a predefined set of inputs, which can then be distributed to a variety of attacks.
APPENDIX

C

LAB TOPOLOGY
Figure C.1 Diagram of Current Lab.